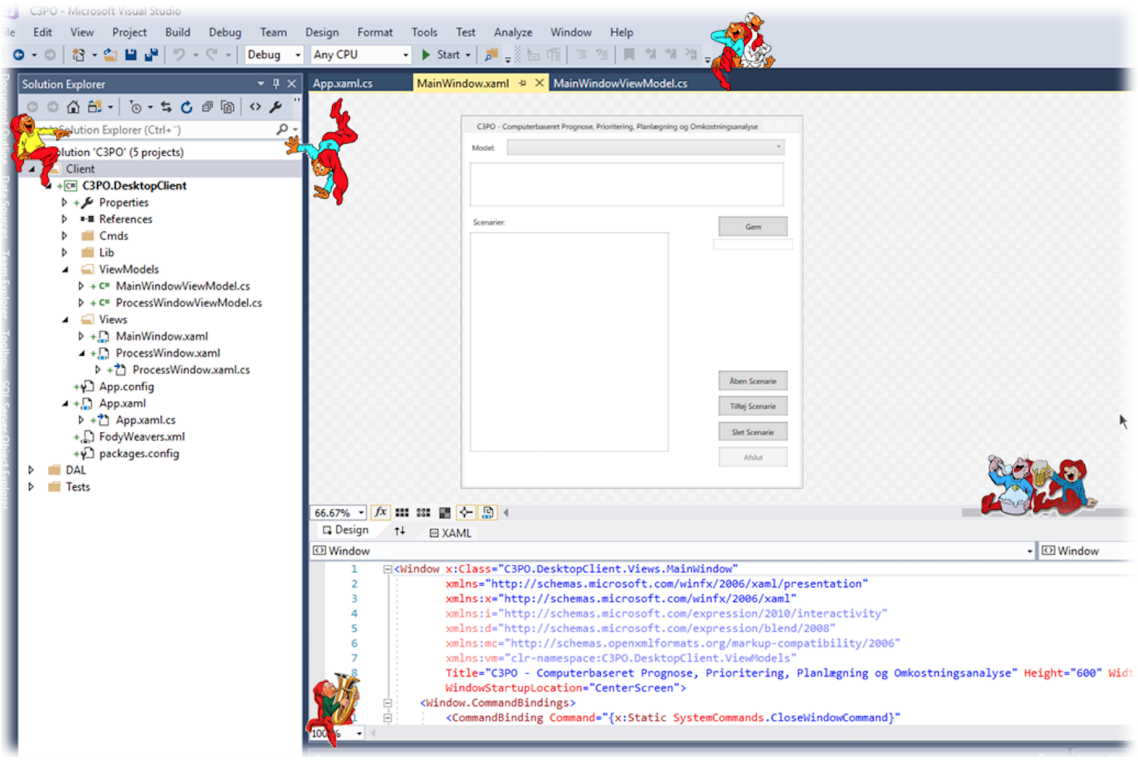
DTU

IT Diplomuddannelsen

Avanceret Objektorienteret Programmering - Efterår 2017



C-3PO: Computerbaseret Prognose, Planlægning, Prioritering og Omkostningsanalyse

Studerende: Allan Køtter
Studienummer: s140381
Underviser: Henrik Tange

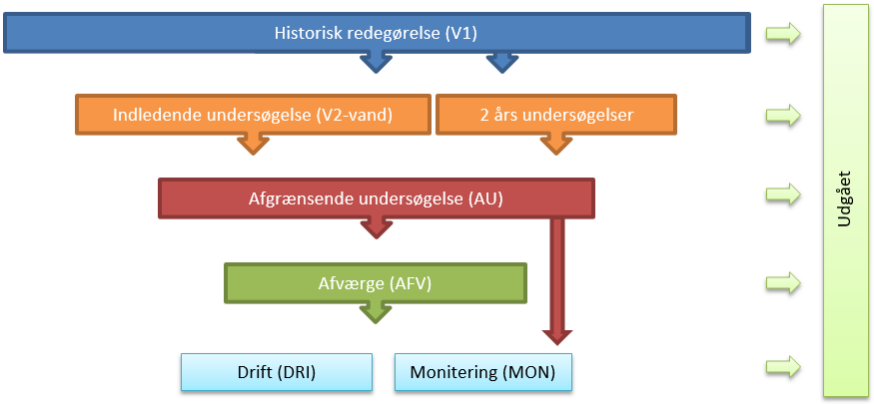
Dato: 19. december 2017
Ordtælling: 3495 ord

Indhold

1. Indledning

I politisk styrede organisationer, har administrationen ofte behov for at kunne redegører for langsigtede konsekvenser af budgetændringer eller ændrede principper for prioritering i forhold til afvikling af kerneopgaven. Hvis kerneopgaven er kompleks, med mange sammenhængende processer og mange hensyn, kan dette være meget svært.

Regionerne gennemfører den offentlige indsats over for jordforurening jf. jordforureningsloven. Målet med indsatsen er at sikre mennesker, miljø og natur mod truslen fra gamle forureninger. Dette sker efter fødekæde-princippet (se figur 1), hvor vidensniveauet om forureningen forædles i hvert led led af fødekæden, og på den baggrund opnås en mere sikker risikovurdering, hvorved flere og flere lokaliteter kan frasorteres. De resterende lokaliteter, der vurderes at kunne udgøre en risiko, videreføres til næste led i fødekæden.



Figur 1: "Fødekæden"

Fødekæden kan betragtes som en generel forretningsproces, med en række sammenhængende aktiviteter, der kan defineres med tid, økonomi og personaleressourcer. I dette projekt ønsker jeg at programmere en applikation, der kan modellere den langsigtede afviklingen af jordforureningssager gennem fødekæden ud fra nogle fastsatte rammebetingelser og prioriteringsprincipper. Administrationen skal kunne kunne bruge applikationen til at opsætte forskellige scenarier med ændrede budgetter og/eller prioriteringsprincipper, for derved bedre at kunne redegører for effekten af påtænkte ændringer.

Formålet med projektet er endvidere at undersøge applikationsudvikling med WPF og Entity Framework i en professionel kontekst - dvs. en lagdelt arkitektur efter MVVM principperne.

Selvstudie: Datamodellering med Data Annotations og Fluent API i Entity Framework (EF)

2. Analyse og design

2.1 Kravspecifikation

Applikationen skal bestå af en grafisk brugerflade, hvor brugeren kan oprette nye modelscenarier for en given domænemodel og tilgå en liste over gemte scenarier, samt køre disse. De gemte scenarier skal kunne redigeres af scenarieejeren, og nye scenarier oprettes med default værdier fra et basisscenarie, og skal derefter kunne ændres af scenarieejeren. Et scenarie skal kunne konfigureres af brugeren som en række sammenkoblede aktiviteter, der hver især omfatter et sæt rammebetingelser herunder tid, økonomi og bemanding og et sæt prioriteringsregler.

Når modellen køres af en brugeren, skal applikationen genererer tidsserier for økonomiforbrug, forbrug af mandskabstimer og sager i kø. Tidsserierne skal kunne eksporteres til MS Excel til videre visualisering og præsentation. Applikationens implementering af fødekæden skal være så generel, at forretningsprocesser fra andre domæner/fagområder let skal kunne implementeres.

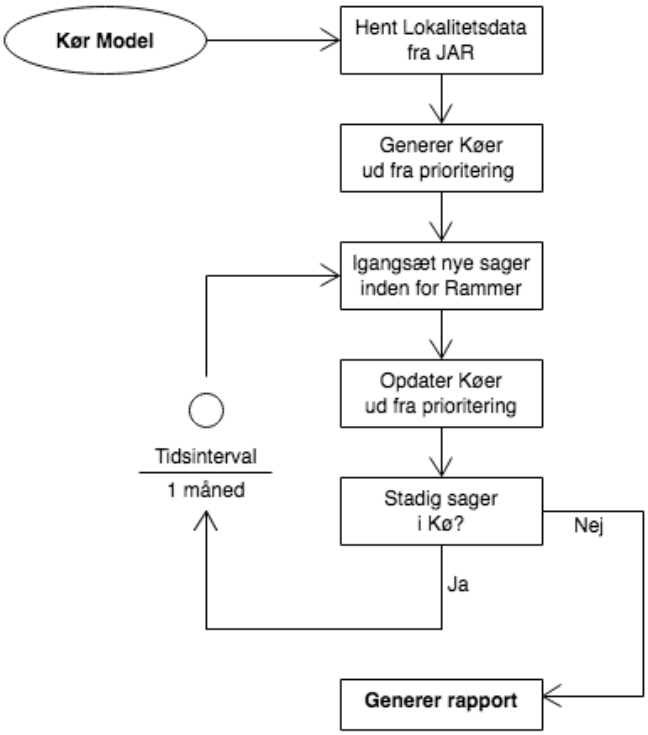
Modelkørslen

Når en model bliver kørt sker der først en indledende initialisering: der laves et totaludtræk fra lokalitetsdatabasen som datagrundlag. Derefter oprettes der køer, hvor hver lokalitet fordeles på et af ledene i fødekæden. Køen sorteres i forhold til de opsatte prioriteringsbetingelser. Herefter er den egentlige kørsel klar til at blive gennemført.

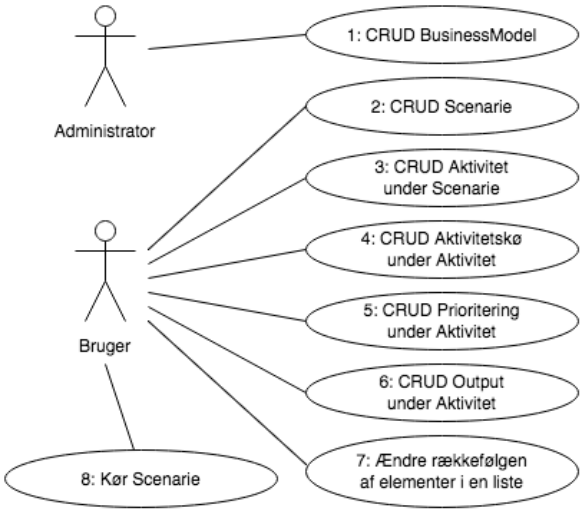
Modelkørslen (figur 2) består af en række iterationer af tidsintervaller på en måned. For hver måned sættes det antal sager i gang på hvert led i fødekæden, som rammebetingelserne tillader herunder budget og personaleressourcer. Når et antal sager sættes i gang, reserveres økonomi og ressourcer til at gennemføre disse, inden for indeværende års budget. Igangværende sagers ressourceforbrug (økonomi og personale) samt gennemførte sager i indeværende måned registreres. Tilsidst beregnes nye køer for hvert led i fødekæden, ved at trække igangsatte sager fra og lægge hitrater for gennemførte sager til næste led i fødekæden.

2.2 Use Cases

Ud fra kravspecifikationen kan følgende Use Cases identificeres (tabel 1 og figur 3). For overblikket skyld er de Use Cases, der omfatter handlingerne *Opret, Vis, Vis Alle, Ret, og Slet* for samme objekt, samlet i en grupperet *"CRUD-Usecase"*. Nedenstående Usecasediagram viser dermed ikke isolerede usecases, som UML ellers foreskriver.



Figur 2: Flowdiagram, der illustrerer modelkørslen i applikationen



Figur 3: Use Case diagram med 8 Use Cases. Use Cases er oplistet i tabel 1 og er nærmere beskrevet i bilag C

Tabel 1: Use case oversigt med reference til domænemodel

ID	Navn	Domæne	Kompleksitet	Prioritet
1	CRUD BusinessModel	Model	Høj	Lav
2	CRUD Scenarie	Scenarie	Høj	Høj
3	CRUD Aktivitet	Aktivitet	Høj	Høj
4	CRUD Aktivitetskø	Aktivitet	Høj	Mellem
5	CRUD Prioritering	Prioritering	Høj	Lav
6	CRUD Output	Output	Høj	Lav
7	Ændre rækkefølgen af elementer i en liste	Aktivitet og Prioritering	Mellem	Mellem
8	Kør Scenarie	Scenarie	Høj	Høj

2.3 Implementering af forretningslogik

Implementering af forretningslogikken sker ved, at der i applikationen kan oprettes en *BusinessModel* (forretningsmodel), der repræsenterer en domænespecifik forretningsproces f.eks. fødekæden. En forretningmodel indeholder en liste af *processer*, der er brugerdefinerede scenarier for afviklingen af forretningsprocessen. En forretningsmodel indeholder endvidere generelle informationer og definitioner, der er fælles for alle scenarierne, herunder enheder, tidsdiskredisering for modellens afvikling og modellerede parametre.

Datagrundlaget for en modelkørsel vil typisk være et datasæt, der afspejler den aktuelle status for igangværende og afventende *"enheder"*, der skal igennem processen. Datasættet vil være relateret til den domænespecifikke forretnigsproces, men være ens for alle scenarier. Der implementeres derfor en datadefinition i forretningsmodellen i form af en connectionstring og en SQL sætning, hvorved et opdateret datasæt kan hentes forud for en modelkørsel.

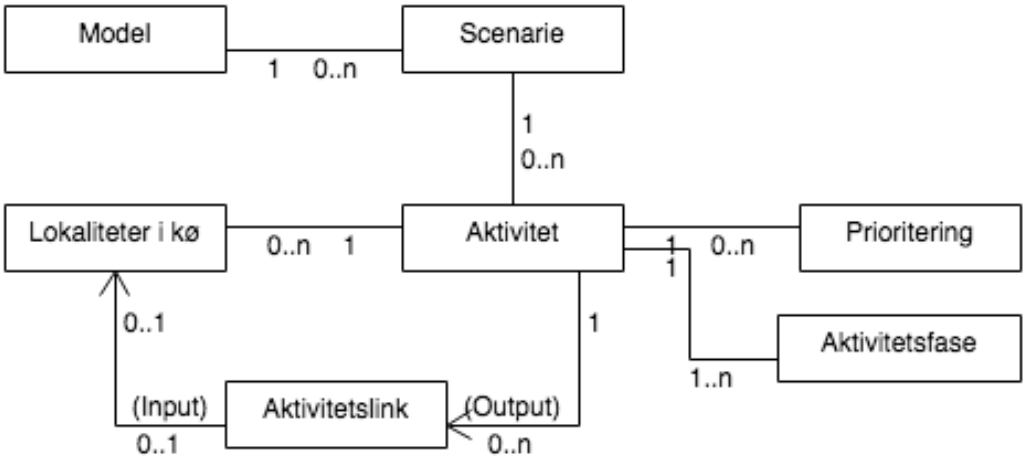
Hver brugerdefineret processcenarie vil indeholde en liste af *aktiviteter*, hvor hvert led i fødekæden kan konfigureres med et sæt rammebetingelser, der bl.a. omfatter budget, personaleressourcer, enhedsomkostninger og sagsbehandlingstid. Disse elementer betragtes som rammebetingelser, fordi de sætter en begrænsning for, hvor mange enheder der kan komme igennem aktiviteten pr. tidsenhed. Hver aktivitet indeholder også en kø til afventende enheder.

Endvidere indeholder en aktivitet en liste med evt. prioriteringsregler. En prioriteringsregel baseres på en rækkefølge af værdier i en given liste f.eks. kommunenavne eller risikokoder. Prioriteringsregler implementeres for at kunne imødekomme lovbundne regler eller politiske værdier som f.eks. "værdifulde grundvandsområder er vigtigst"eller "boligundersøgelser skal tages først".

Aktiviteterne i et scenarie skal kunne kobles sammen ved at definerer et eller flere output fra en aktivitet, således at uotput'et fra aktivitet bliver til input'et i en anden. I sammenkoblingen defineres en sandsynlighed baseret på erfaringsmæssige hitrater. Den samlede sandsynlighed for alle output under en akivitet skal være 100%.

2.4 Domænemodel

Domænemodellen er vist på figur 4.



Figur 4: Domænemodel for C-3PO. Domænerne er refereret i tabel 1

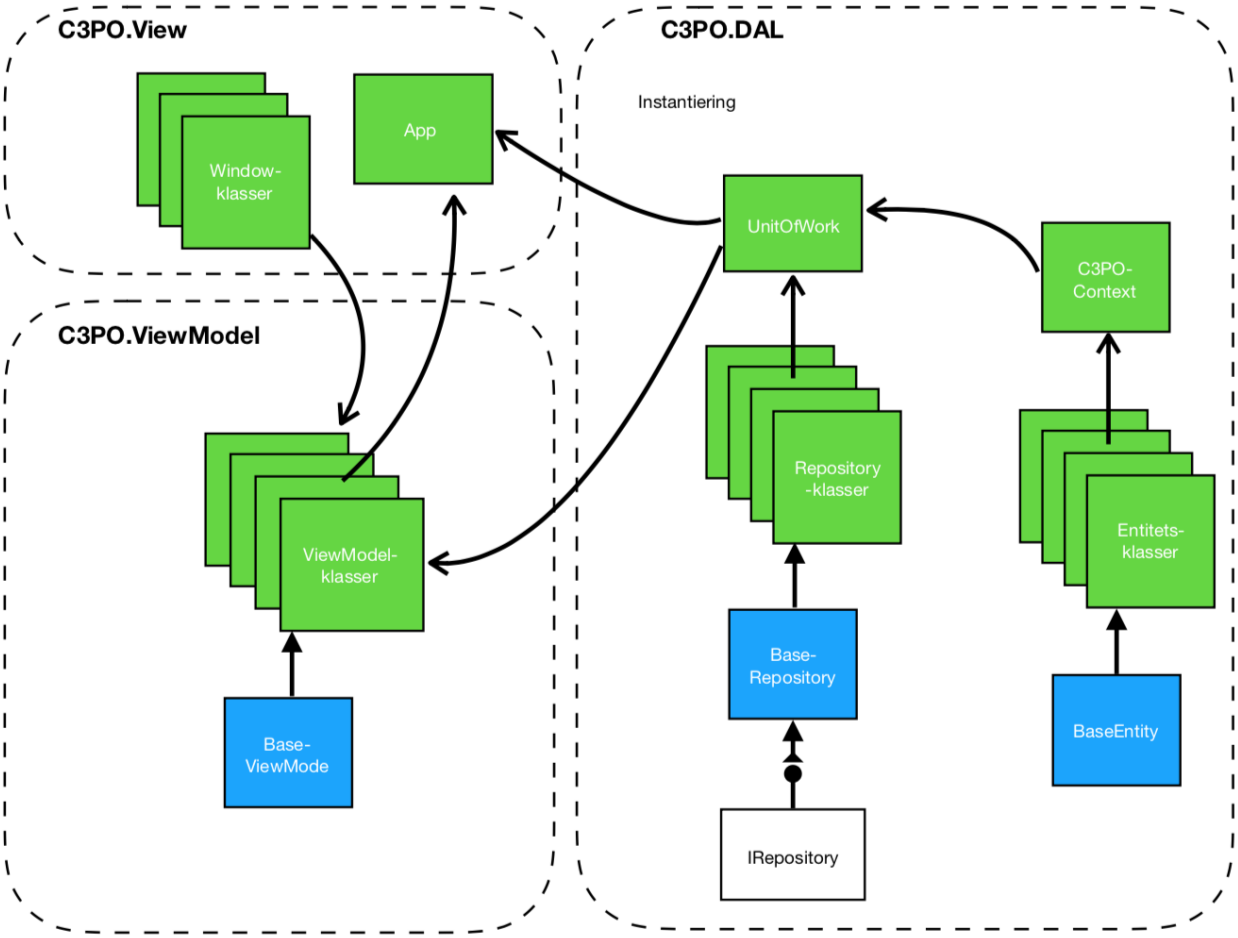
2.5 Overordnet applikations arkitektur

Applikationen udvikles ud fra et 3-lags princip, der opdeles i et datalag (*Data Access Layer*), et applikationslag hvor al forretningslogik implementeres (ViewModel), og endelig et præsentationslag, der indeholder den grafiske brugerflade. Fordelen ved denne seperation af data, forretningslogik og præsentation er, uddeligering af ansvar, hvilket er i overensstemmelse med det objektorienterede paradigme. Endvidere lettes vedligeholdelsen af applikationen, da det enkelte lag kan opdateres eller udskiftes, uden at skulle omprogrammere hele applikationen.

Datalaget programmeres ved hjælp af *Entity Framework* (EF), som er en *ORM* (Object Relational Mapper) i .NET frameworket. EF er den mest anvendte teknologved udvikling af professionelle database-applikationer i dag. I datalaget implementeres endvidere *Repository*-mønsteret samt *Unit Of Work*, for at have let adgang til at udfører CRUD-operationer mod databasen.

Selce præsentationslaget udvikles vha. WPF-frameworket. For at opnå en klar seperation af applikations- og præsentationslaget, implementeres *MVVM*-principperne (Model-View-ViewModel), da dette også er standarten i dag ved udvikling af professionelle WPF-applikationer. Snitfladerne *INotifyPropertyChanged*, *INotifyError* og *INotifyDataErrorInfo* implementeres i forretningslager og i datalaget, for lettere at opnå en responsiv brugerflad, med tovejs databinding og feltvalidering. Den overordnede arkitektur er skitseret på figur 2.5.

Klassediagrammer for de enkelte lag findes i bilag D



Figur 5: Applikations arkitektur for C3PO (skitse)

3. Bemærkninger til kodens indhold og opbygning

3.1 Entitets-klasserne og DbContext

Selve datamodellen er implementeret i applikationen med Entity Framework (EF) og *Code First* (se kapitel 5). Med EF opnås mapningen mellem entitets-klasserne og databasen ved hjælp af C3POContext-klassen, der nedarver fra DbContext. Mapningen sker ved instantiering af et DbSet-objekt for hver entitet/tabel [11].

Det er muligt at arbejde direkte med DbSet-objekterne ved at instantiere C3POContext-klassen, men dette er ikke hensigtsmæssigt, bl.a. fordi evt. tabel-specifik datafunktionalitet f.eks. sql-forespørgsler der anvendes mange steder i applikationen, ville skulle gentages. Derfor er der implementeret repository-klasser for hver DbSet, der "wrapper"al kommunikation med DbSet-objekterne [1].

3.2 Implementering af *Repository* mønsteret og UnitOfWork

Repository mønsteret er implementeret som en del af Datalaget (DAL), for at give let adgang til at udføre CRUD-handlinger og anden tabel-specifik funktionalitet mod databasen. Mønsteret er implementeret med snitfladen IRepository, der definere de CRUD-properties, der skal indgå i alle repository-klasserne. Derefter er der programmeret en base-klasse, der implementerer abstrakte properties af de pågældende CRUD-properties, undtagen *Delete*-properties, der kræver en specifik instantiering af de respektive DbSet, for at kunne gennemføre slettehandlinge.

Endelig er der oprettet specifikke repository-klasser for hver DbSet, der nedarver baseklassen og endvidere implementerer de "manglende"*Delete*-properties i base-klassen. Repository-klasserne instantieres med en instans af C3POContext for at få adgang til databasen og DbSet-objekterne i denne.

Unit Of Work klassen er oprettet, for at opnå en entydig adgang til repository-klasserne og for at instantiere repository-klasserne med samme C3POContext-objekt. Med andre ord - når man opretter en instans af UnitOfWor-klassen, har man automatisk adgang til alle repository-klasserne [6].

3.3 Implementering af INotifyPropertyChanged

Snitfladen *INotifyPropertyChange* giver mulighed for at opnå en interaktiv brugerflade, idet WPF frameworkets kontroller indeholder triggere, der kan abbonerer på Event'et *PropertyChanged*. Som udgangspunkt har jeg implementeret snitfladen på baseklasse *Proper-*

tyChanged, og ladet Entitets-klasserne og ViewModel-klasserne nedarve denne.

Dette er kommet til at virke, men det er en stor og tidskrævende opgave at indsætte metoden *NotifyPropertyChanged()* i alle properties i alle klasserne, hvilket er nødvendigt for at præsentationslagets kontroller kan abbonerer på ændringer i klassernes properties. Jeg har derfor valgt at installere *PropertyChange.Fody*, og dermed implementere NotifyPropertyChange() metoden som en "FodyWeaver", der anvender injection på kompileringstidspunktet.

3.4 Implementering af interaktiv datavalidering i brugerfladen

Snitfladerne *IDataErrorInfo* og *INotifyDataErrorInfo* giver mulighed for at implementere interaktiv validering af felter i WPF. Snitfladerne er implementeret direkte i mine ViewModel-klasser, hvorved metoden *HasErrors()* og eventet *ErrorsChanged* bliver tilgængelige, og præsentationslagets kontroller kan bindes til disse.

Der er programmeret en hjælpeklasse, der har metoder til at udfører generelle valideringscheck, og returnerer sand eller falsk. Der er kun implementeret en metode på nuværende tidspunkt som "proof-of-concept", der validerer værdien i tekstfelter, der ikke må være tomme. Tanken er at udbygge denne klasse med andre metoder, og flytte den til et generlisk kodebibliotek, der fremadrettet kan genanvendes i andre projekter.

Fody frameworket tillbyder også en færdigstrikket løsning der hedder *Validar.Fody*, der implementerer de to snitflader som en "FodyWeaver"på somme måde som PropertyChange.Fody. Jeg har ikke undersøgt denne mulighed nærmere, men overvejer at gøre dette i det videre udvikling af applikationen.

3.5 Lagdelt arkitektur

Applikationen blev oprindelig programmeret som 3 projekter (C3PO.DAL.dll, C3PO.ViewModel.dll og C3PO.DesktopClient.exe) samt 3 test-projeter. Pga. udfordringer med cirkulær reference mellem ViewModel og DesktopClient, er disse 2 projekter samlet i ét projekt (DesktopClient.exe). Dette er udelukkende for at kunne demonstrerer den grafiske brugerflade til eksamen, og når problematikken omkring cirkulær reference er løst, bliver projektet igen opdelt i 3 eller flere lag.

4. Test

Test Driven Development (TDD) er afprøvet og undersøgt som en del af projektet med *MSTest Framework* og Visual Studios *Test Explorer*. Formålet med TDD er at sikre, at den ønskede funktion implementeres med mindst mulig kode, og samtidig sikre, at funktionaliteten fortsat virker, når applikationen videreudvikles / ændres.

TDD sker ved at man starter med at oprette et test til den funktionalitet, man ønsker at implementerer. Testen vil fejle, fordi funktionaliteten ikke er implementeret endnu. Derefter implementeres den ønskede funktionalitet i applikationen, og det sikres at testen lykkedes.

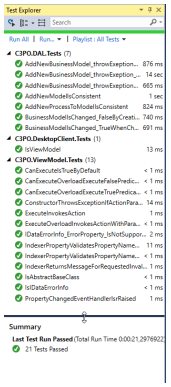
I dette projekt er der oprettet et testprojekt for hver af de 3 projekter C3PO.DAL, C3PO.ViewModel og C3PO.DesktopClient. Testprojeterne er ikke oprettet som en del af applikationens kodestruktur, men er lagt i en seperat folder. Kun udvalgte dele af applikationen er testet med TDD, da TDD tager tid og ikke har være det primære fokus i dette projekt. Gennemførte tests er vist på figur 6

4.1 Unittest

Ved Unittest testes der for isoleret funktionalitet, der ikke har afhængigheder til andre dele af applikationen eller anden funktionalitet. Der er gennemfør Unittests i forbindelse med implementeringen af INotifyPropertyChanged og INotifyDataErrorInfo.

4.2 Funktionstest

Modsat unittests har funktionstest afhængigheder til andre dele af programmet. Der er gennemført funktionstest i forbindelse med implementeringen af Entity Frameworks DbContext, hvor forbindelsen til databasen og skrivning til databasen er testet. Disse tests har afhængigheder til Entity Framework og til C3POcontext (implementeringen af DbContext).

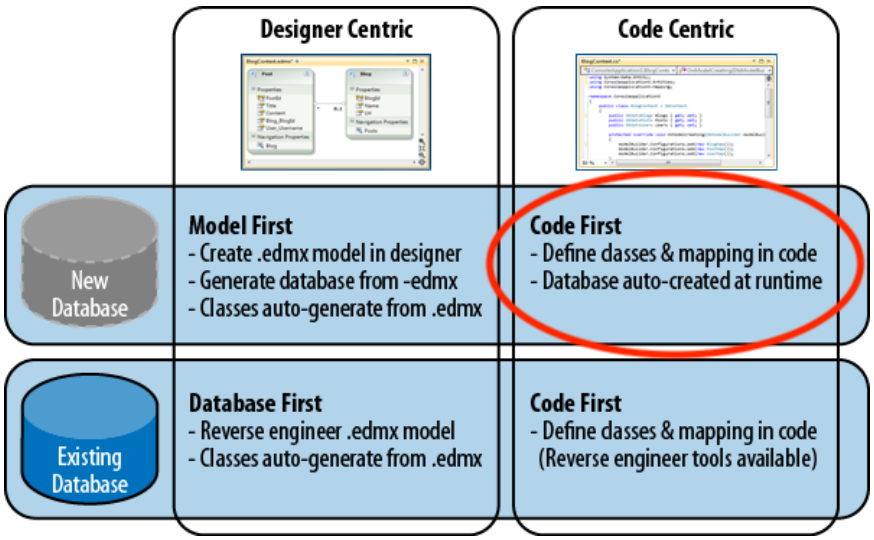


Figur 6: "Screenshot af Visual Studios Test Explorer med gennemførte tests"

5. Selvstudie: Datamodellering med Data Annotations og Fluent API i Entity Framework (EF)

Der findes 2 overordnede arbejdsgange for, at implementerer databaselogik i en applikation med *Entity Framework* (EF) hhv. *Code First* og *Model First / Database First* (herefter benævnt "Model First"). Med Model First tilgangen bruges EF til at autogenererer klasser ud fra en eksisterende datamodel. Dette scenarie anvendes typisk, hvis der udvikles en ny applikation mod en eksisterende database, eller hvor applikationsudviklingen og database-udviklingen varetages af forskellige firmaer, afdelinger eller personer.

Med Code First beskrives datamodellen som en del af kildekoden og nye databaser kan genereres med EF vha. *Migrering*. Code First kan også godt anvendes mod en eksisterende database ved "reverse ingenering"af datamodellen [9, 11]. Kapitlet her vil fokusere på det scenarie, hvor en ny datamodel beskrives vha. C# kode (se fig 7), og jeg vil undersøge graden af kontrol, som programmøren har over datamodellen vha. *Data Annotations* og *the Fluent API* med reference til traditionel SQL .



Figur 7: Illustration af overordnede arbejdsgange i EF (figur fra [9])

5.1 Code First fra et helikopterperspektiv

Code First gør det muligt for programmøren at beskrive entiteter, relationer samt disses egenskaber i datamodellen vha. C# klasser. Den basale datastruktur i modellen detekteres af EF ved vha. af "conventions". Konventionerne er et regelsæt, der er defineret i

namespacet System.Data.Entity.ModelConfiguration.Conventions og som anvendes til en standardiseret oversætte EF klassernes proterties til databasespecifikke modelkonventioner [4].

Code First tager således udgangspunkt i princippet *"convention over configuration"*. Med *Data annotations* i form af attributprogrammering i EF klasserne samt *the Fluent API* har programmøren adgang til at overskrive konventionerne med en specifik konfiguration og dermed overtage kontrollen med oversættelsen bl.a. i forhold til feltlængder, defaultværdier, constraints, index'er, nøglerfelter, tabelreferencer og regler for sletning og opdatering i relaterede tabeller.

Denne kontrol vil ofte være altafgørende for at opnå en effektiv og fintunet datamodellen. Datamodellen er det fundament som applikationen skal hvile på og i en professionel kontekst, vil denne del af applikationsudviklingen være vigtig for f.eks. applikationens performance.

5.2 Attributprogrammering med *Data Annotations*

Programmøren får adgang til Data Annotations med de to namespaces *System.ComponentModel.DataAnnotations* og *System.ComponentModel.DataAnnotations.Schema*, der stiller attributklasser til rådighed, hvormed programmøren kan definere metadata på klassers properties. EF kan anvende disse metadata til datamodel-gennerering.

Primærnøgler

Et af de bærende pricipper i relationelle databaser, er at tabeller indeholde en nøglekolonne, der kan identificere rækkerne i tabellen og dermed bruges som reference i andre tabeller. Konventionen i EF er, at hvis en EF klasses property navngives Id eller *<klassenavn>*Id, så oprettes denne property som primærnøgle i databasetabellen. Programmøren kan overskrive denne konvention, ved at angive attributten [Key] over en eller flere properties.

Felt typer, feltlængder og constraints på feltniveau

Konventionerne omkring oversættelse af felttyper er baseret på, at de fleste .NET typer svarer til en ækvivalent datatype i databasen. Dog giver typen *string* en udfordring, da der ikke ligger nogen længdeinformation i denne type, og *string* oversættes derfor som standart til databasetypen *nvarchar(max)*. De fleste realtionelle databasesystemer har har indbygget længdeinformation i tekstfelter (f.eks. *varchar(25)*) og dette er med til at optimerer databasen væsentligt [5].

Programmøren kan konfigurere sig ud af dette med attributten [StringLength=n], hvor n er et heltal der angiver længden på strengen. Herved oversættes string-datatypen i stedet til *nvarchar(n)*. Som supplement kan der sættes begrænsninger på feltlængden med attributterne [MinLength=n1, Maxlength=n2], hvorved der oprettes constraints for dette i databasen. EF implementerer samtidig automatisk check af disse i f.eks. WPF vinduer. Yderligere kan programmøren konfigurerer oversættelsen af en property med attributten [Column]. [Colunm] attributten tager 3 parametre: Navn, Order og TypeName, hvormed

konventionerne for feltnavnet, feltets index i tabellen og felttype kan sættes f.eks. [Column("MitFeltNavn", Order=3, TypeName="varchar")].

Et væsentligt element i datamodellering er, at angive hvilke felter der kræves udfyldt og hvilke der ikke kræves udfyldt. Som standart anvender EF en konvention om at der oprettes "NOT NULL"kolonner for klassernes Id properties og properties med værdi-datatyper. Reference-datatyper oprettes tilgengæld alle som "NULL"kolonner. Programmøren kan konfigurerer en reference-datatype som "NOT NULL"med attributten [Required] [5].

Defaultværdier for properties kan konstrueres med C# kode på klasseniveau, men hvis det er en del at logikken i datamodellen, kan programmøren vælge at lade databasen generer værdierne. Dette kan opnås med attributten [DatabaseGenerated(DatabaseGenerationOption.Computed)], hvorved at property'en mappes til en beregnet kolonne i tabellen og bliver read-only. [DatabaseGenerated(DatabaseGenerationOption.Computed)] anvendes ofte på primærnøgler, hvor databasen har ansvaret for at tildele en værdi, når en ny række indsættes i tabellen, men kan også bruges ved tildeling af f.eks. oprettelsesdato eller systembruger ved indsættelse af en ny række.

Relationer og Foreign Keys

Konventionen omkring relaterede tabeller er, at hvis en klasse indeholder en liste af en anden klasses objekter og denne klasse indeholder en property med "forældre objektet", så oversættes dette som en-til-mange relation i databasemodellen, hvorved der oprettes en fremmedenøgle i den relaterede tabel-definition. Som standart oprettes en fremmednøgle med samme navn som Id-kolonnen i forældre-klassen. Med [ForeignKey] attributten kan programmøren angive, at en property med et andet navn, skal oprettes som fremmednøgle i tabellen.

Konventionen kan oversætte simple relationer, men kan ikke identificere mere komplekse relationer. F.eks. en klasse, der skaber en relation mellem to objekter fra den samme klasse, vil ikke blive korrekt oversat af EF konventionerne. Programmøren kan her anvende [InverseProperty(<*property*>)] attributten, for at konfigurere sådan en relation, så den bliver korrekt implementeret i databasen.

Det implementeres ved at der i relationsklassen oprettes en property til hver af de to objekter der skal relateres. Samtidig oprettes der to liste-properties i de objekter, der skal relateres, der kan indeholde de relationsobjekter, som de indgår i. Attributten [InverseProperty(<*property*>)] sættes over disse to liste-properties, hvor <*property*> refererer til den property i relations-klassen, som objektet er oprettet i [5].

Properties der ikke skal indgå i databasemodellen

Programmøren har mulighed for at markerer, at en property ikke skal indgå i datamodellen med attributten [NotMapped]. Dette kan anvendes hvis entitetsklasserne indeholder properties, der udelukkende anvendes til applikations-relaterede forretningslogik i runtime, men ikke ønskes lagret i databasen.

5.3 The Fluent API

I langt de fleste tilfælde vil brugen af Data Annotations være tilstrækkelig for at kunne kontrollerer de fleste datamodeller. Dog vil der være tilfælde, hvor Data Annotations ikke

slår til og her giver The Fluent API mulighed for en endnu større kontrol over oversættelsen. The Fluent API tilgås ved at overstyre metoden OnModelCreating i DbContext klassen.

Et eksempel på en konfiguration, der ikke kan opnås med Data Annotations, er konfiguration af mange-til-mange relationer mellem to tabeller. Konfiguration i the Fluent API sker metoderne *.HasMany<Entitet1>(e1 => e1.Property).WithMany(e2 => e2.Property)*

Fjernelse af ON DELETE CASCADE constraints på relationer er et andet eksempel på en configuration, som ikke kan opnås med Data annotations. Konfigurationen i the Fluent API opnås med metoden .WillCascadeOnDelete(false)

5.4 Opsummering

Datamodellen bag en applikation vil være det fundament, som resten af applikationen byggerpå, og derfor et centralt element i applikationsudvikling. *Data Annotations* giver programmøren mulighed for en stor grad af kontrol over datamodellering, i forbindels med Entity Framework's *Code First* funktionalitet. Med *The Fluent API* kan programmøren opnå endnu mere finkornet kontrol over datamodellen, og kan konfigurerer datamodellen på samme detaljeret niveau som i SQL.

Disse muligheder bør derfor anvendes, hvis en applikation udvikles med afsæt i *Code First* paradigmet. Hvis der udvikles med en eksisterende datamodel, hvor datamodellering ikke falder inden for programmørens ansvarsområde, bør der i stedet tages afsæt i Entity framewok's *Database first*, og *Data Annotations / The fluent API* vil ikke være aktuelle.

6. Konklusion

Målet med projektet har været at udvikle en applikation der kan give administrationen i Region Hovedstaden mulighed for at modellere forskellige scenarier for færdiggørelsen af den offentlige jordforureningsopgave. En scenariekørsel skal kunne give administrationen indsigt i konsekvenser af ændrede budgetter og/eller prioriteringsprincipper, for derved bedre at kunne redegører for effekten af evt. påtænkte ændringer.

Projektet er blevet gennemført ved at programmere en Windows-applikation i .NET baseret på WPF frameworket og med database-understøttelse baseret på Entity Framework. Målet med projektet har desuden været at undersøge applikationsudvikling med en lagdelt arkitektur efter MVVM principperne.

6.1 Resultat

Applikationen er på nuværende tidspunkt ikke færdigudviklet, men hele fundamentet omkring den lagdelte arkitektur er implementeret, herunder et fungerende Data Access Layer (DAL), et fungerenede applikationslag (Viewmodel) der understøttet intaaktiv databinding og validering i præsentationslaget (View). I præsentationslaget er der oprettet to sammenhængende vinduer, der udnytter interaktiv binding til applikationslaget og giver mulighed for at gemme data i databasen.

Jeg har endvidere i dette projekt udforsket mulighederne for datamodellering med *Data Annotations* og *The Fluent API*, samt undersøgt og afprøvet Test Driven development (TDD).

6.2 Perspektivering

Pga. valget og et nyt politisk udvalg på miljøområdet i Region Hovedstaden, skal der i foråret 2018 gennemføres strategiarbejde bl.a. på jordforureningsområdet. Politikerne skal her præsenteres for en række scenarier for færdiggørelsen af jordforureningsopgaven. C3PO applikationen skal derfor gøres færdig, så scenarierne kan modelleres og konsekvenserne af politikernes valg f.eks. besparelser synliggøres.

Selve den lagdelte applikationsarkitektur med implementering af MVVM mønsteret, har desuden givet mig et godt fundament, til at komme videre med .NET. Jeg forventer f.eks. at mit næste læringsmål inden for .NET programmering bliver webprogrammering med ASP.NET, hvor den lagdelte applikations arkitektur også kan anvendes. Hvis jeg ønskede at videreudvikle C3PO til en webapplikation, forventer jeg i høj grad at kunne genbruge data- og applikationslaget uden væsentlige ændringer og dermed primært at kunne fokusere på udviklingen af et webbaseret præsentationslag.

Endvidere ønsker jeg at grave dybere i begraberne *Test Driven Development* og *"Clean Architecture"*, og inddrage disse udviklingsstrategier i min værktøjskasse.

Referencer

[1] David Anderson. *Enterprise MVVM*. 2014. URL: https://www.youtube.com/user/dcomnetwork.

[2] Krzysztof Cwalina og Brad Abrams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET*. 2nd. Addison-Wesley, 2008. ISBN: 978-0-321-54561-9; 0-321-54561-3.

[3] *Entity Framework Development Workflows*. Microsoft Developer Network. 2016. URL: https://msdn.microsoft.com/en-us/library/ee712907(v=vs.113).aspx.

[4] *Entity Framework (EF) Documentation*. Microsoft Developer Network. 2016. URL: https://msdn.microsoft.com/en-us/library/ee712907(v=vs.113).aspx.

[5] *Entity Framework Tutorial*. 2017. URL: http://www.entityframeworktutorial.net/.

[6] Mosh Hamedani. *Repository Pattern with C and Entity Framework, Done Right*. 2015. URL: https://www.youtube.com/user/programmingwithmosh.

[7] Microsoft. *Choosing Between Class and Struct*. Udg. af Microsoft Developer Network. 2009. URL: https://msdn.microsoft.com/en-us/library/ms229017.aspx.

[8] Microsoft. *Lambda Expressions (C# Programming Guide)*. Udg. af Microsoft Technical Documentation. 2017. URL: https://docs.microsoft.com/en-us/dotnet/articles/csharp/programming-guide/statements-expressions-operators/lambda-expressions.

[9] Julia Miller Rowan ; Lerman. *Programming Entity Framework: Code First*. O'Reilly Media, Inc., 2011.

[10] Benjamin Perkins, Jacob Vibe Hammer og Jon D Reid. *Beginning C# 6 Programming with Visual Studio 2015*. 1st. Birmingham, UK, UK: Wrox Press Ltd., 2015. ISBN: 1119096685, 9781119096689.

[11] Andrew Troelsen og Philip Japikse. *C# 6.0 and the .NET 4.6 Framework*. 7th. Apress, 2015. ISBN: 9781484213339.

[12] *UMLetino. A free online UML tool for fast UML diagrams*. 2017. URL: http://www.umlet.com/umletino/umletino.html.

Bilag

A. Anvendte værktøjer

Applikationen er udviklet ved brug af følgende værktøjer:

- Visual Studio Community 2017
  - Entity framework v.6
  - PropertyChanged.Fody (https://github.com/Fody/PropertyChanged)
  - MSTest Framework og Test Explorer.
- UMLetino, A free online UML tool for fast UML diagrams [12].

Rapporten er skrevet med MacTex (Latex distribution til Mac) med håndtering af referencer i Bibtex.

B. Brugervejledning

Åben og luk applikationen

Applikationen åbnes ved at køre programmet C3PO.desktopClient.exe, hvorved startvinduet vises (se figur 8). Med knappen "Afslut"i startvinduet kan applikationen afsluttes.

Startvinduet

Startvinduet repræsenterer en model af en forretningsproces. I dropdown-boksen øverst i vinduet starter man med at vælge den model, som man ønsker at arbejde med. Når en model er valgt, vises en beskrivelse af modellen i tekstfeltet nedenfor, og allerede oprettede processscenarier for modellen, vises i listen.

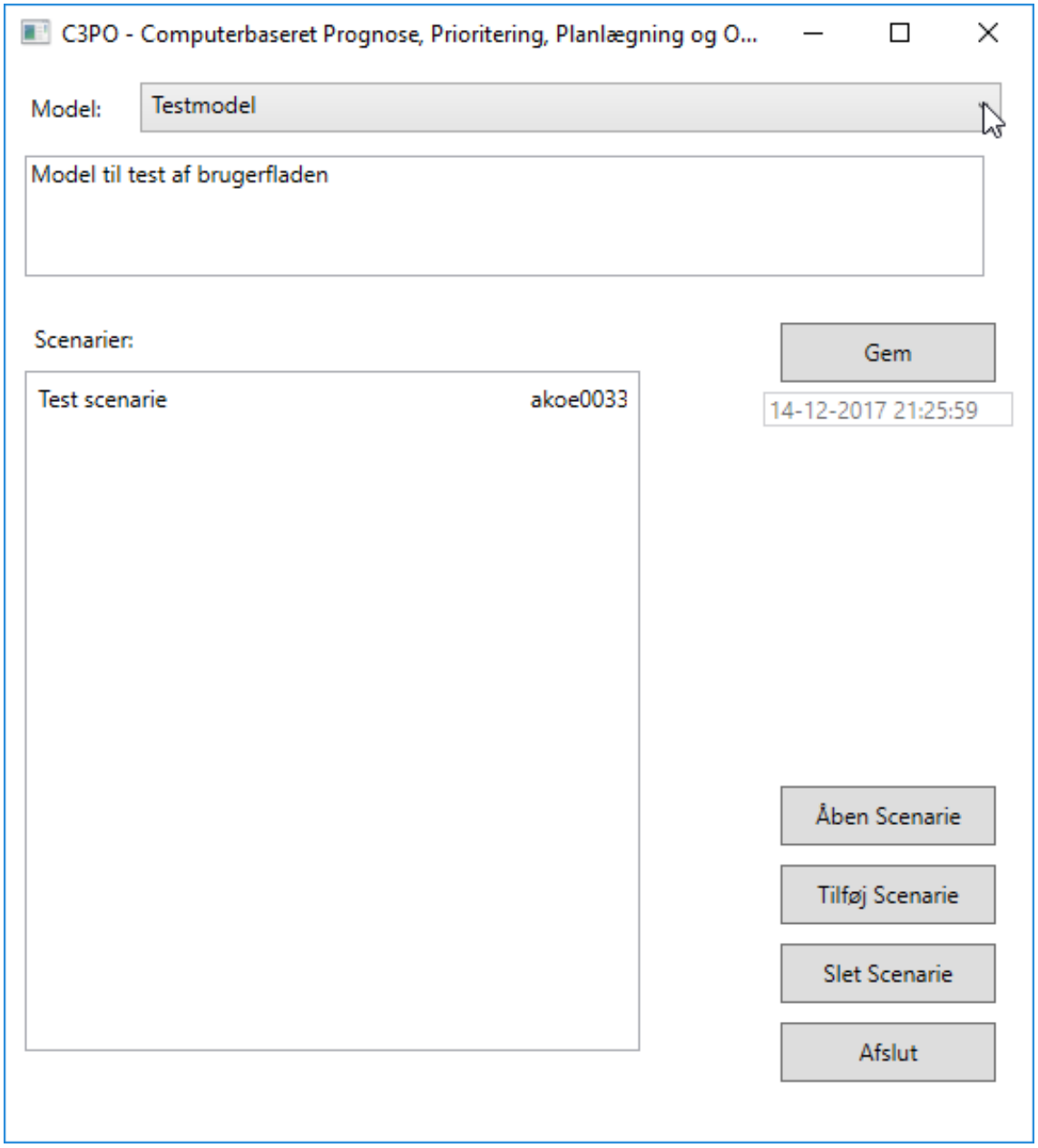
Beskrivelsen kan editeres (figur 9), og ændringerne kan gemmes ved at trykke på knappen "Gem". Tidsfeltet nedenunder knappen afspejler, hvornår seneste ændring er gemt i databasen, og kan benyttes til at verificerer, at ændringerne man har lavet bliver gemt (figur 10). Knapperne "Åben scenarie", "Tilføj scenarie"og "Gem scenarie"fungerer ikke implementeret endnu.

Scenarie-vinduet

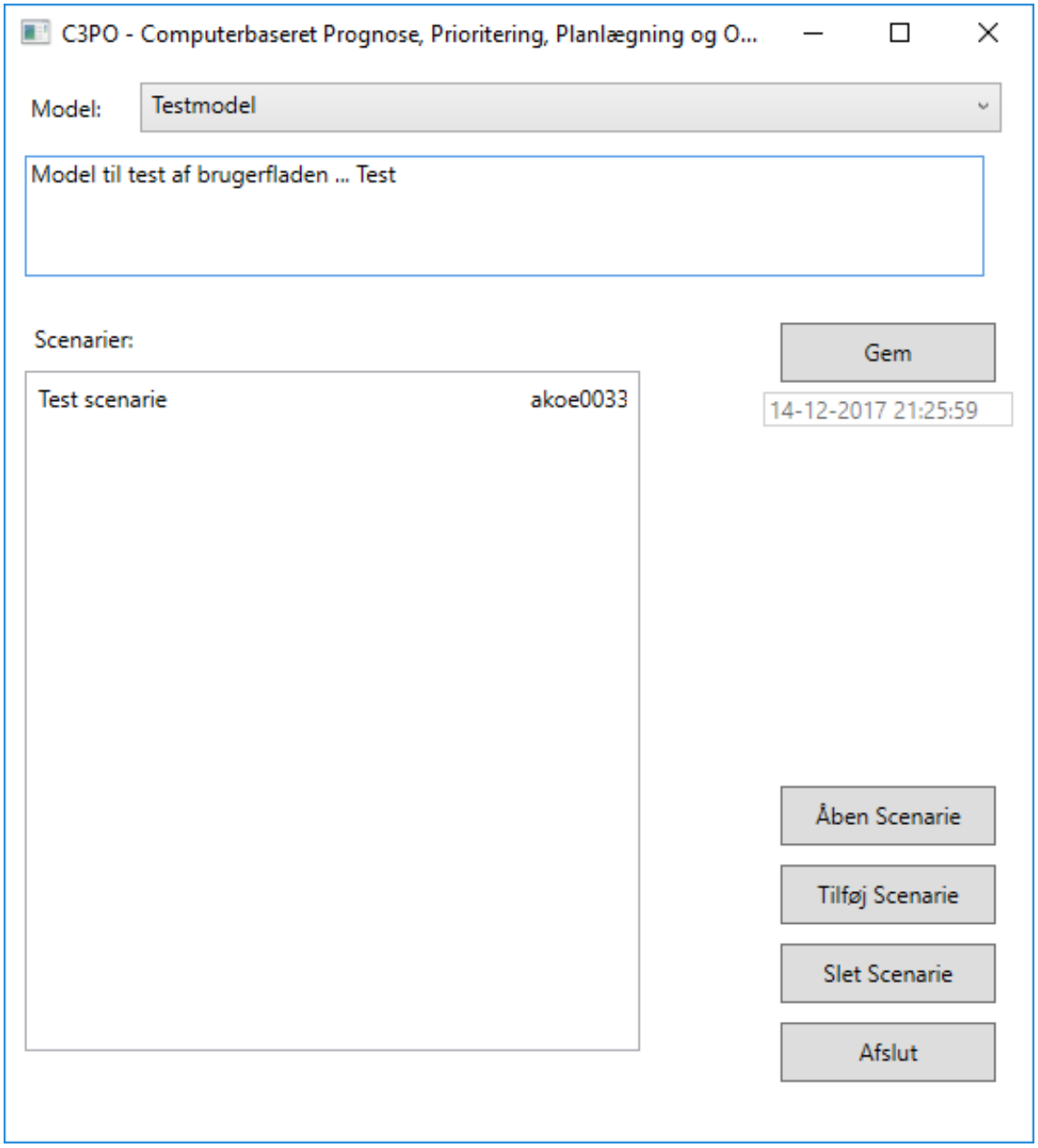
Dobbelt-klikkes på et scenarie i startvinduets scenarie-liste (figur 11), åbnes det pågældende scenarie i et nyt scenarie-vindue (figur 12). Basisdata for scenariet vises øverst i vinduet - navn og beskrivelse kan editeres, mens felterne "Ejer", "Oprettet"og "Sidst opdateret"automatisk opdateres af applikationen. "Ejer"og "Oprettet"fastsættes ved oprettelsen af et nyt scenarie, mens "Sidst opdateret"opdateres, når ændringer til et scenarie gemmes. Ændringer kan gemmes til databasen med knappen "Gem"

Aktivitets-listen i scenariet viser de aktiviteter, der er oprettet under et scenarie. Knapperne "Tilføj Aktivitet", "Ret Aktivitet"og "Slet aktivitet"er ikke implementeret endnu. Knapperne "Flyt op"og "Flyt ned"under "Sortering", knytter sig til Aktivitets-listen, og er heller ikke implementeret endnu. Knappen "Kør Scenarie"er heller ikke implementeret endnu.

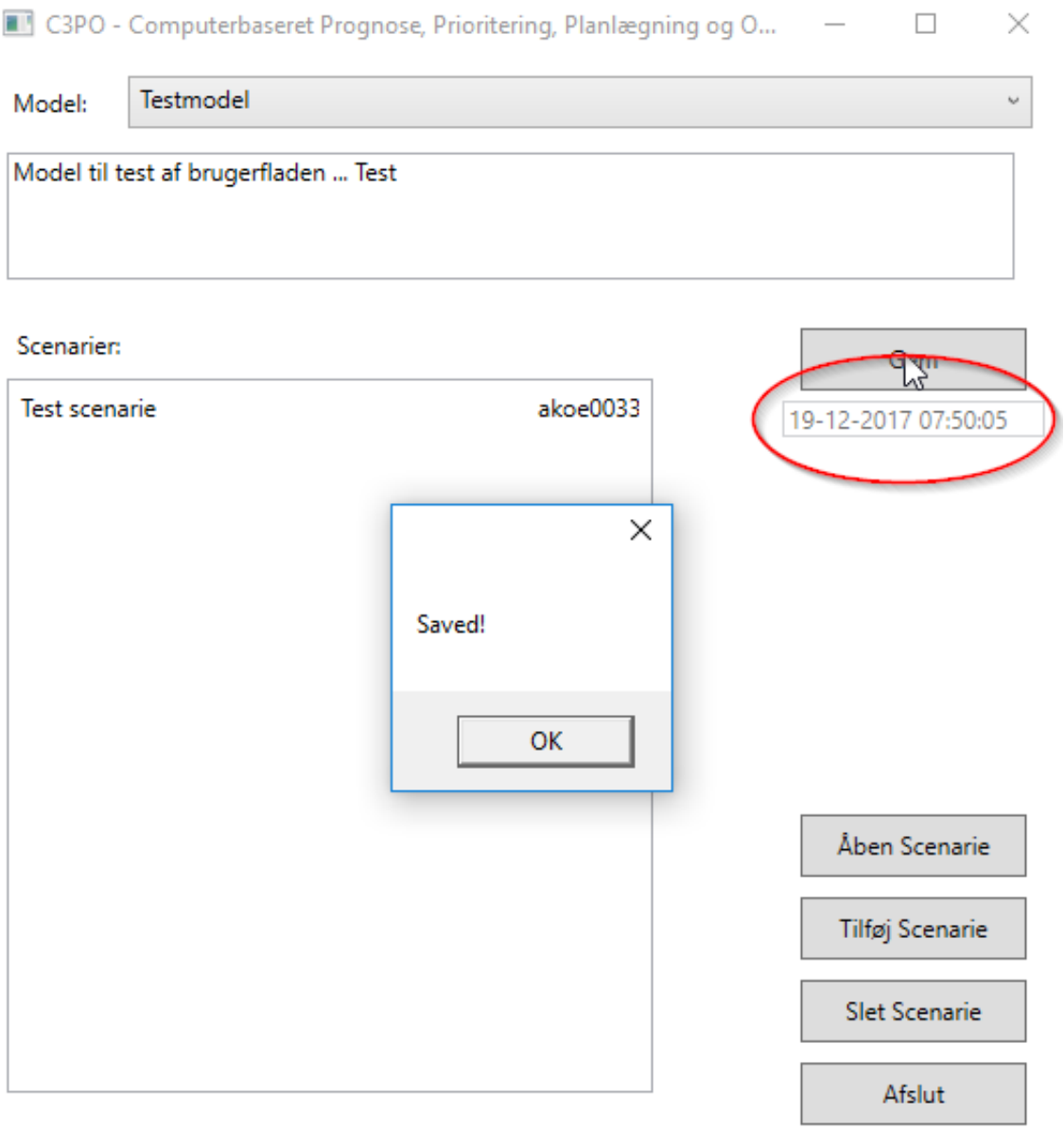
Vinduet lukkes med knappen "Luk".



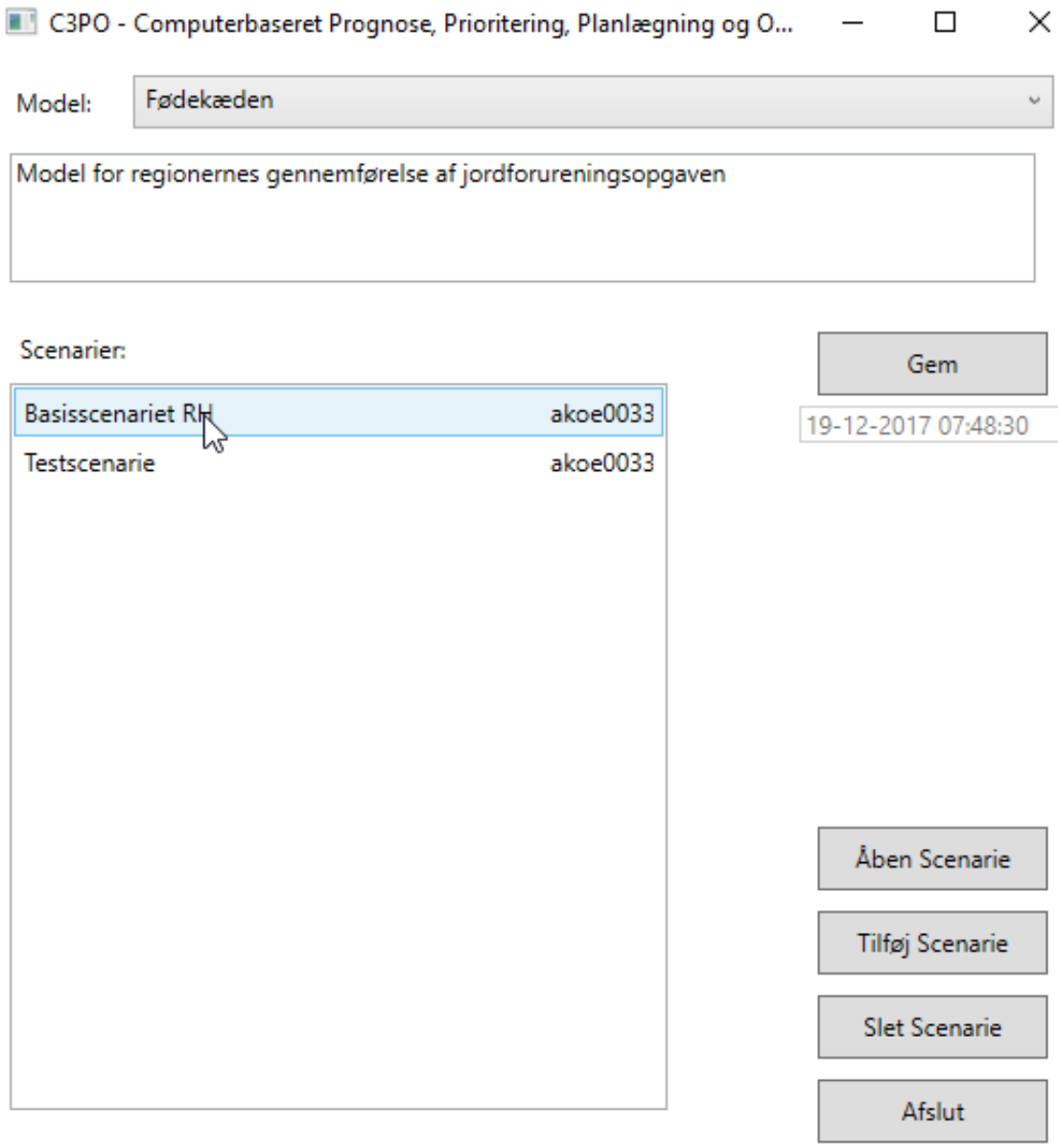
Figur 8: Applikationens startvinduet



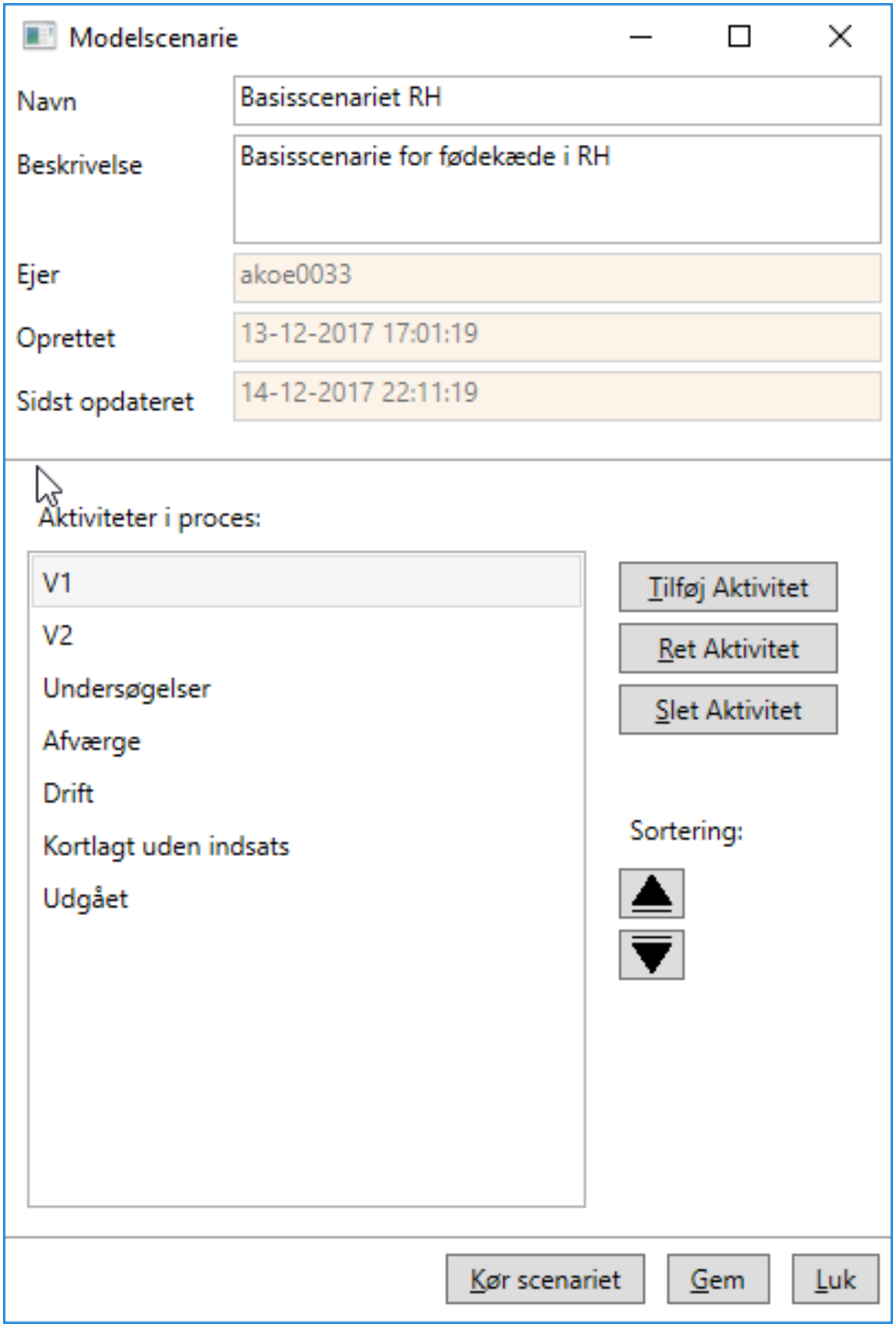
Figur 9: Editering af modellens bemærkningsfelt



Figur 10: Ændringer gemmes til databasen



Figur 11: Åbning af scenarie ved dobbeltklik i scenarie-listen



Figur 12: Scenarievinduet, med scenariets data indlæst

C. Use Cases

UC1: CRUD BusinessModel

Aktør: Bruger

Betingelser:

Succes kriterier: Bruger kan oprette, vise, redigere og slette modeller i systemet jf CRUD paradigmet.

Primær arbejdsgang: Opret

1. Bruger åbner applikationen og startvindue vises.
2. Bruger klikker på knap med tandhjulsikon i øverste højre hjørne af startvinduet.
3. Et nyt Administrations vindue åbner med en liste indeholdende gemte modeller i applikationen.
4. Bruger vælger "Ny"for at oprette en ny model.
5. Modelvinduet åbnes.
6. Brugeren udfylder felterne i vinduer.
7. Bruger vælger Gem
8. Modellen gemmes

Alternativ arbejdsgang 1: Vis

1. Bruger åbner applikationen og startvindue vises.
2. Bruger klikker på knap med tandhjulsikon i øverste højre hjørne af startvinduet.
3. Et nyt Administrations vindue åbner med en liste indeholdende gemte modeller i applikationen.
4. Brugeren dobbeltkliker på en model i listen
5. Et modelvinduet åbnes med data for den valgte model.
6. Bruger vælger Luk
7. Vinduet lukker uden ændringer i data

Alternativ arbejdsgang 2: Rediger

1. Bruger åbner applikationen og startvindue vises.
2. Bruger klikker på knap med tandhjulsikon i øverste højre hjørne af startvinduet.
3. Et nyt Administrations vindue åbner med en liste indeholdende gemte modeller i applikationen.
4. Brugeren markerer en model og klikker på "rediger"knappen
5. Et modelvinduet åbnes med data for den valgte model.

6. Brugeren retter oplusningerne i de ønskede felter.
7. Bruger vælger Gem
8. Modellen gemmes med de ændrede data

Alternativ arbejdsgang 3: Slet

1. Bruger åbner applikationen og startvindue vises.
2. Bruger klikker på knap med tandhjulsikon i øverste højre hjørne af startvinduet.
3. Et nyt Administrations vindue åbner med en liste indeholdende gemte modeller i applikationen.
4. Bruger vælger "Slet"for at slette en eksisterende model.
5. Popup vindue: "Ønsker du virkelig at slette denne model"
6. Bruger vælger "Ja"
7. Modelle slettes og fjernes fra listevinduet

UC2: CRUD Scenarie

Aktør: Bruger

Betingelser: Brugeren har åbnet applikationern og valgt den model han/hun ønsker at arbejde med i dropdown listeøverst i vinduet. Gemte scenarier under modellen vises i scenarie-listen.

Succes kriterier: Bruger kan oprette, vise, redigere og slette scenarier i systemet jf CRUD paradigmet.

Primær arbejdsgang: Opret

1. Bruger klikker på knappen "Tilføj".
2. et nyt scenarie oprettes baseret på basis scenariet.
3. Scenarie vinduet åbnes med data fra det nye scenarie og med brugerens navn i Bruger feltet.
4. Brugeren retter evt. ønskede felter.
5. Bruger vælger Gem
6. Det nye scenarie gemmes

Alternativ arbejdsgang 1: Vis

1. Brugeren dobbeltkliker på et scenarie i listen
2. Et scenarie vinduet åbnes og med data for det valgte scenarie.
3. Bruger vælger Luk
4. Vinduet lukker uden ændringer i data

Alternativ arbejdsgang 2: Rediger

1. Brugeren markerer et scenarie i listen og klikker på "rediger"knappen
2. Et Scenarie vindue åbnes med data for det valgte scenarie.
3. Brugeren retter oplysningerne i de ønskede felter.
4. Bruger vælger Gem

5. Scenariet gemmes med de ændrede data

Alternativ arbejdsgang 3: Slet

1. Brugeren markerer et scenarie i listen og vælger "Slet".
2. Popup vindue vises: "Ønsker du virkelig at slette?"
3. Bruger vælger "Ja"
4. Scenariet slettes og fjernes fra listevinduet

UC3: CRUD Aktion

Aktør: Bruger

Betingelser: Brugeren har åbnet applikationern og valgt den model han/hun ønsker at arbejde med i dropdown listeøverst i vinduet. Gemte scenarier under modellen vises i scenarie-listen. Brugeren har valgt et scenarie i listen og klikket på knappen "Rediger". Et scenarievindue er blevet åbnet, med data for et valgte scenarie og en liste med scenariets gemte aktiviter.

Succes kriterier: Bruger kan oprette, vise, redigere og slette aktiviteter under et scenarie jf CRUD paradigmet.

Primær arbejdsgang: Opret

1. Bruger klikker på knappen "Tilføj".
2. Et Aktivitet vinduet åbnes med tomme felter.
3. Brugeren udfylder nødvendige felter og klikker på knappen "Gem"
4. Vinduet lukkes og den nye aktivitet er blevet tilføjet til aktivitetslisten

Alternativ arbejdsgang 1: Vis

1. Brugeren dobbeltkliker på et scenarie i listen
2. Et scenarie vinduet åbnes og med data for det valgte scenarie.
3. Bruger vælger Luk
4. Vinduet lukker uden ændringer i data

Alternativ arbejdsgang 2: Rediger

1. Brugeren markerer et scenarie i listen og klikker på "rediger"knappen
2. Et Scenarie vindue åbnes med data for det valgte scenarie.
3. Brugeren retter oplysningerne i de ønskede felter.
4. Bruger vælger Gem
5. Scenariet gemmes med de ændrede data

Alternativ arbejdsgang 3: Slet

1. Brugeren markerer et scenarie i listen og vælger "Slet".
2. Popup vindue vises: "Ønsker du virkelig at slette?"
3. Bruger vælger "Ja"
4. Scenariet slettes og fjernes fra listevinduet

...

(Øvrige CRUD usecases mangler at blive beskrevet)

UC7: Ændre rækkefølgen af elementer i en liste

Aktør: Bruger

Betingelser: Brugeren har åbnet applikationern og har åbnet et vindue der indeholder en liste med elementer i en given rækkefølge.

Succes kriterier: Bruger kan ændre rækkefølgen ved at flytte et element op eller ned i listen.

Primær arbejdsgang: Flyt op

1. Brugeren markerer et element i listen og klikker på "Pil op"knappen
2. Det valgte element flytter et trin op i listen.
3. Bruger vælger Gem.
4. Den valgte rækkefølge for elementerne gemmes.

Alternativ arbejdsgang 1: Flyt ned

1. Brugeren markerer et element i listen og klikker på "Pil ned"knappen
2. Det valgte element flytter et trin ned i listen.
3. Bruger vælger Gem.
4. Den valgte rækkefølge for elementerne gemmes.

UC8: Kør Model

Aktør: Bruger

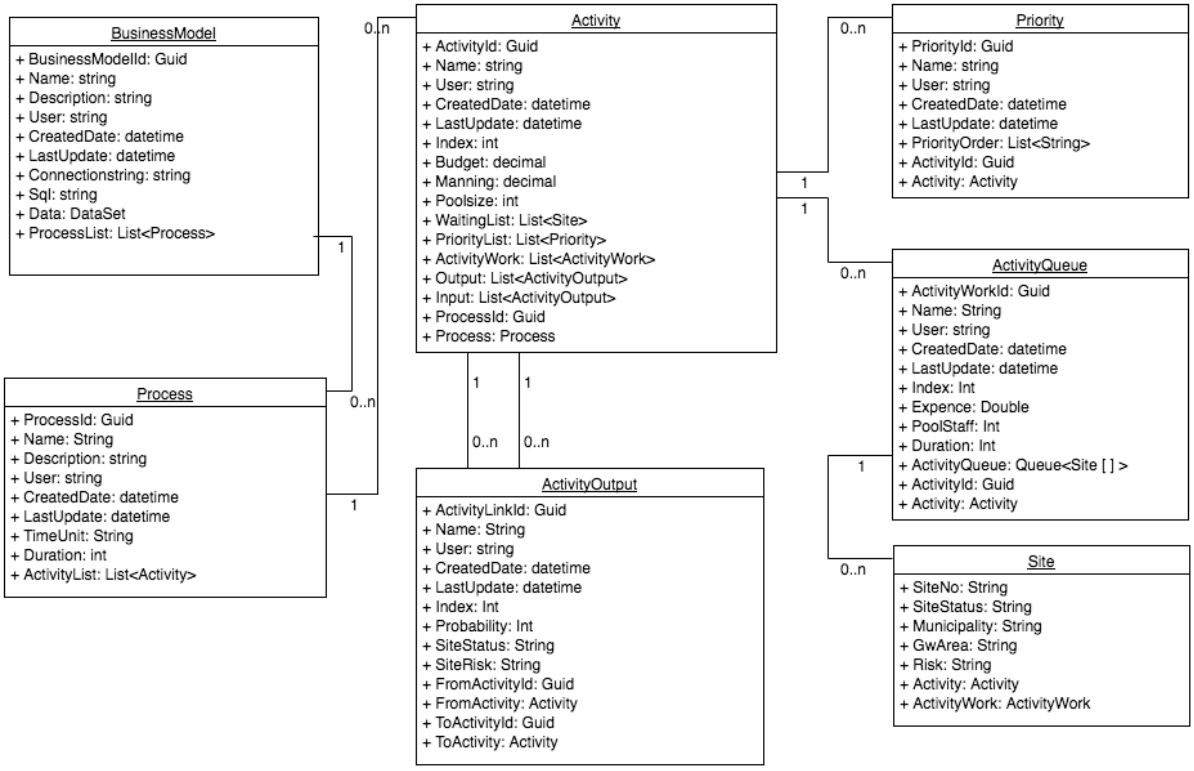
Betingelser: Bruger står i et scenarievindue med et gemt scenarie.

Succes kriterier: Bruger kan lave en modelkørsel og generer en rapport med modelresultater

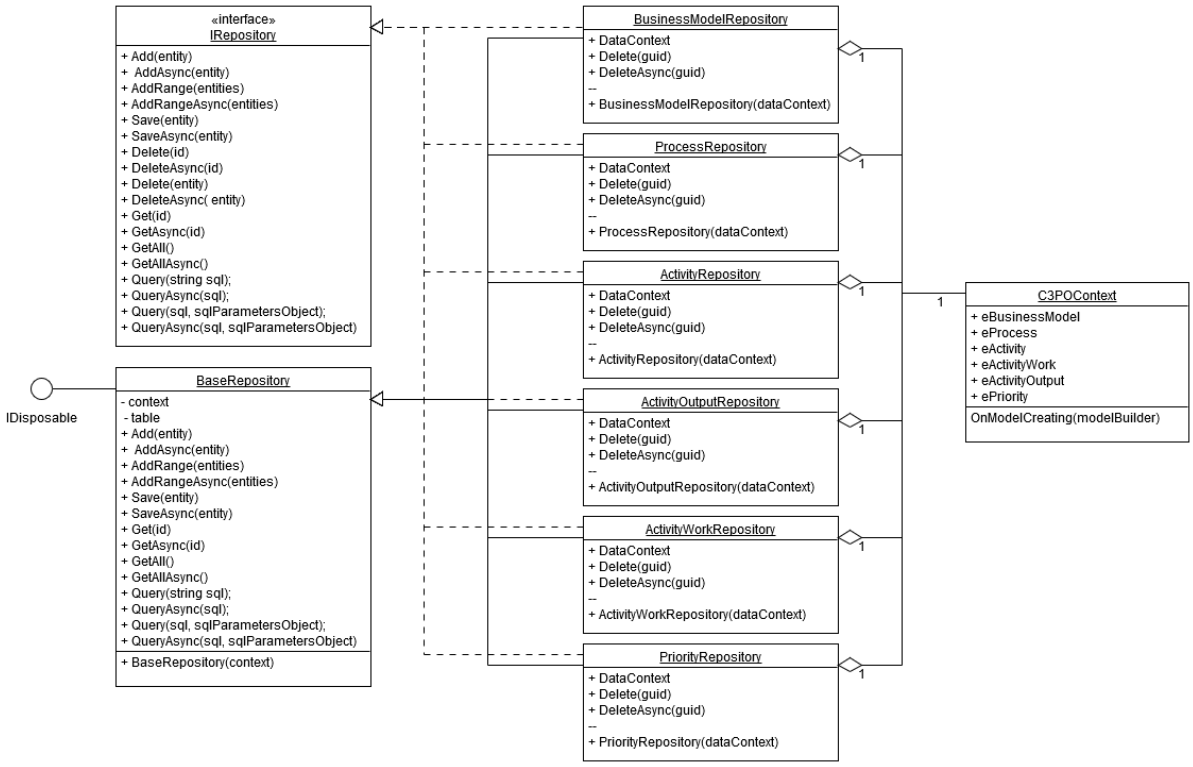
Primær arbejdsgang:

1. Bruger vælger "Kør".
2. Et snurrende "progress wheel"vises mens modellen kører.
3. Efter modelkørslen er tilendebragt, vises et resultat-vindue, med opsummerende statusinformation for kørslen.
4. Brugeren vælger "Eksporter til Excel"
5. Excel åbner med data fra modelkørslen i form af tidsserier
6. Bruger lukker resultatvinduet. Resultat vinduet lukker.

D. Klassediagrammer



Figur 13: Klassediagram for Entitetsklasser i DAL



Figur 14: Klassediagram for Repositoryklasser i DAL

E. Udskrift af koden: C3PO.DAL

```
 1  using System;
 2  using System.Collections;
 3  using System.Collections.Generic;
 4  using System.Collections.ObjectModel;
 5  using System.Linq;
 6  using System.Text;
 7  using System.Threading.Tasks;
 8  using System.ComponentModel;
 9  using System.ComponentModel.DataAnnotations;
10  using System.ComponentModel.DataAnnotations.Schema;
11  using PropertyChanged;
12
13  namespace C3PO.DAL.Lib
14  {
15
16      public abstract class BaseModel : IDataErrorInfo, INotifyDataErrorInfo, INotifyPropertyChanged
17      {
18          [Timestamp]
19          public byte[] Timestamp { get; set; }
20
21          [Required, DataType(DataType.DateTime)]
22          public DateTime CreatedDate { get; set; } = DateTime.Now;
23
24          [DataType(DataType.DateTime)]
25          public DateTime LastUpdate { get; set; } = DateTime.Now;
26
27          #region "Property Changed"
28          public event PropertyChangedEventHandler PropertyChanged;
29          [NotMapped]
30          public bool IsChanged { get; set; } = false;
31
32  #endregion
33
34          #region "IDataerrorInfo"
35          // IDataErrorInfo
```

```
36
37          [NotMapped]
38          public virtual string this[string columnName]
39          {
40              get { return OnValidate(columnName); }
41          }
42
43          [NotMapped]
44          public string Error
45          {
46              get
47              {
48                  throw new NotSupportedException();
49              }
50          }
51
52          protected virtual string OnValidate(string propertyName)
53          {
54              var context = new ValidationContext(this)
55              {
56                  MemberName = propertyName
57              };
58
59              var results = new Collection<ValidationResult>();
60              var isValid = Validator.TryValidateObject(this, context, results, true);
61
62              if (!isValid)
63              {
64                  ValidationResult result = results.SingleOrDefault(p => p.MemberNames.Any(memberName => memberName ==
                      propertyName));
65                  return result == null ? null : result.ErrorMessage;
66              }
67
68              return null;
69
```

```
70          }
71
72          protected string[] GetErrorsFromAnnotations<T>(string propertyName, T value)
73          {
74              var results = new List<ValidationResult>();
75              var vc = new ValidationContext(this, null, null)
76              { MemberName = propertyName };
77              var isValid = Validator.TryValidateProperty(value, vc, results);
78
79              return (isValid) ? null : Array.ConvertAll(
80                  results.ToArray(), o => o.ErrorMessage);
81          }
82          #endregion
83
84          #region "INotifyDataErrorInfo
85          //INotifyDataErrorInfo
86          protected readonly Dictionary<string, List<string>> Errors =
87              new Dictionary<string, List<string>>();
88
89          protected void ClearErrors(string propertyName = "")
90          {
91              Errors.Remove(propertyName);
92              OnErrorsChanged(propertyName);
93          }
94
95          protected void AddError(string propertyName, string error)
96          {
97              AddErrors(propertyName, new List<string> { error });
98          }
99
100         protected void AddErrors(string propertyName, IList<string> errors)
101         {
102             var changed = false;
103             if (!Errors.ContainsKey(propertyName))
104             {
```

```
105                 Errors.Add(propertyName, new List<string>());
106                 changed = true;
107             }
108             errors.ToList().ForEach(x =>
109             {
110                 if (Errors[propertyName].Contains(x)) return;
111                 Errors[propertyName].Add(x);
112                 changed = true;
113             });
114             if (changed)
115             {
116                 OnErrorsChanged(propertyName);
117             }
118         }
119         public IEnumerable GetErrors(string propertyName)
120         {
121             if (string.IsNullOrEmpty(propertyName))
122             {
123                 return Errors.Values;
124             }
125             return Errors.ContainsKey(propertyName) ? Errors[propertyName] : null;
126         }
127 
128         public bool HasErrors => Errors.Count != 0;
129 
130         public event EventHandler<DataErrorsChangedEventArgs> ErrorsChanged;
131 
132         protected void OnErrorsChanged(string propertyName)
133         {
134             ErrorsChanged?.Invoke(this, new DataErrorsChangedEventArgs(propertyName));
135         }
136 #endregion
137 
138     }
139 
```

```
140  }
141
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.ComponentModel.DataAnnotations;
7  using System.ComponentModel.DataAnnotations.Schema;
8  using C3PO.DAL.Lib;
9
10 namespace C3PO.DAL.Models
11 {
12     [Table("eBusinessModel")]
13     public partial class BusinessModel: BaseModel
14     {
15         // Properties
16         [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
17         public Guid ModelId { get; set; }
18         [Required, StringLength(50)]
19         public string Name { get; set; }
20         [MaxLength]
21         public string Description { get; set; }
22         [Required, MaxLength]
23         public string ConnectionString { get; set; }
24         [Required, MaxLength]
25         public string Sql { get; set; }
26
27         public virtual List<Process> ProcessList { get; set; } = new List<Process>();
28
29
30
31     }
32 }
33
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Collections.ObjectModel;
7  using System.ComponentModel.DataAnnotations;
8  using System.ComponentModel.DataAnnotations.Schema;
9  using C3PO.DAL.Lib;
10
11 namespace C3PO.DAL.Models
12 {
13     [Table("eProcess")]
14     public partial class Process : BaseModel
15     {
16         // Properties
17         [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
18         public Guid ProcessId { get; set; }
19         [Required, StringLength(50)]
20         public string Name { get; set; }
21         [Required, MaxLength]
22         public string Description { get; set; }
23         [Required, StringLength(10)]
24         public string User { get; set; } = Environment.UserName;
25
26
27         public virtual List<Activity> ActivityList { get; set; } = new List<Activity>();
28
29         // Foreign Key property
30         [Required, ForeignKey("Model"), Column(Order = 0)]
31         public Guid ModelId { get; set; }
32
33         // Navigational properties
34         public virtual BusinessModel Model { get; set; }
35
```

```
36      }
37  }
38
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Collections.ObjectModel;
7  using System.ComponentModel.DataAnnotations;
8  using System.ComponentModel.DataAnnotations.Schema;
9  using C3PO.DAL.Lib;
10
11 namespace C3PO.DAL.Models
12 {
13     [Table("eActivity")]
14     public partial class Activity : BaseModel
15     {
16         // Properties
17         [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
18         public Guid ActivityId { get; set; }
19         [Required, StringLength(50)]
20         public string Name { get; set; }
21         [Required]
22         public int Index { get; set; }
23         [Required]
24         public int Budget { get; set; }
25         [Required]
26         public int Staff { get; set; }
27         [Required]
28         public int Poolsize { get; set; }
29         [Required]
30         public float PoolsPrPerson { get; set; }
31
32
33         public virtual List<Priority> Priorities { get; set; } = new List<Priority>();
34         public virtual List<Activitywork> Workflow { get; set; } = new List<Activitywork>();
35         [InverseProperty("InputActivity")]
```

```
36          public virtual List<Activitylink> Inputs { get; set; } = new List<Activitylink>();
37          [InverseProperty("OutputActivity")]
38          public virtual List<Activitylink> Outputs { get; set; } = new List<Activitylink>();
39
40          // Foreign Key property
41          [Required, ForeignKey("Process"), Column(Order = 0)]
42          public Guid ProcessId { get; set; }
43
44          // Navigational properties
45          public virtual Process Process { get; set; }
46
47          [NotMapped]
48          public virtual List<Site> WaitingList { get; set; } = new List<Site>();
49
50
51      }
52  }
53
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Collections.ObjectModel;
7  using System.ComponentModel.DataAnnotations;
8  using System.ComponentModel.DataAnnotations.Schema;
9  using C3PO.DAL.Lib;
10
11
12 namespace C3PO.DAL.Models
13 {
14     [Table("rActivityLink")]
15     public partial class Activitylink : BaseModel
16     {
17         // Properties
18         [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
19         public Guid ActivitylinkId { get; set; }
20         [Required, StringLength(50)]
21         public string Name { get; set; }
22         [Required]
23         public int Index { get; set; }
24         [Required]
25         [Range(0, 1)]
26         public float Probability { get; set; }
27         [Required, StringLength(50)]
28         public string SiteStatus { get; set; }
29         [Required, StringLength(6)]
30         public string SiteRisk { get; set; }
31
32         // Foreign Key property
33         [ForeignKey("InputActivity")]
34         public Guid? InputActivityId { get; set; }
35
```

```
36          [ForeignKey("OutputActivity")]
37          public Guid? OutputActivityId { get; set; }
38
39          // Navigational properties
40          [ForeignKey("InputActivityId")]
41          public virtual Activity InputActivity { get; set; }
42          [ForeignKey("OutputActivityId")]
43          public virtual Activity OutputActivity { get; set; }
44
45
46      }
47  }
48
```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using C3PO.DAL.Lib;


namespace C3PO.DAL.Models
{
    [Table("eActivitywork")]
    public partial class Activitywork : BaseModel
    {
        // Properties
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public Guid WorkId { get; set; }
        [Required, StringLength(50)]
        public string Name { get; set; }
        [Required]
        public int Index { get; set; }
        [Required]
        public float PoolExpence { get; set; }
        [Required]
        public float PoolStaff { get; set; }
        [Required]
        public int Duration { get; set; }

        public virtual Queue<Site[]> WQueue { get; set; } = new Queue<Site[]>();

        // Foreign Key property
        [Required, ForeignKey("Activity"), Column(Order = 0)]
        public Guid ActivityId { get; set; }
```

```
36
37          // Navigational properties
38          public virtual Activity Activity { get; set; }
39
40
41      }
42  }
43
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Collections.ObjectModel;
7  using System.ComponentModel.DataAnnotations;
8  using System.ComponentModel.DataAnnotations.Schema;
9  using C3PO.DAL.Lib;
10
11 namespace C3PO.DAL.Models
12 {
13     [Table("ePriority")]
14     public partial class Priority
15     {
16         // Properties
17         [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
18         public Guid PriorityId { get; set; }
19         [Required, StringLength(50)]
20         public string Column { get; set; }
21         [Required]
22         public int Index { get; set; }
23         [Required]
24         public virtual List<String> Order { get; set; } = new List<String>();
25
26         // Foreign Key property
27         [Required, ForeignKey("Activity"), Column(Order = 0)]
28         public Guid ActivityId { get; set; }
29
30         // Navigational properties
31         public virtual Activity Activity { get; set; }
32
33         public void ChangePriority(int index, string direction)
34         {
35
```

```
36          }
37      }
38  }
39
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using C3PO.DAL.Lib;
7
8  namespace C3PO.DAL.Models
9  {
10     public class Site
11     {
12         // Properties
13         public Guid SiteId { get; private set; }
14         public string SiteNo { get; private set; }
15         public string Municipality { get; private set; }
16         public string GwArea { get; private set; }
17         public string SiteStatus { get; set; }
18         public string Risk { get; set; }
19         public virtual Activity Activity { get; set; }
20
21         public Site(Guid siteid, string siteno, string municipality, string gwarea, string status, string risk, Activity
             activity)
22         {
23             this.SiteId = siteid;
24             this.SiteNo = siteno;
25             this.Municipality = municipality;
26             this.GwArea = gwarea;
27             this.SiteStatus = status;
28             this.Risk = risk;
29             this.Activity = activity;
30
31         }
32
33         public void Update(string status, string risk, Activity activity)
34         {
```

```
35              this.SiteStatus = status;
36              this.Risk = risk;
37              this.Activity = activity;
38          }
39      }
40  }
41
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7
8  namespace C3PO.DAL.Lib
9  {
10     interface IRepository<T>
11     {
12         int Add(T entity);
13         Task<int> AddAsync(T entity);
14         int AddRange(IList<T> entities);
15         Task<int> AddRangeAsync(IList<T> entities);
16         int Save(T entity);
17         Task<int> SaveAsync(T entity);
18         int Delete(Guid id);
19         Task<int> DeleteAsync(Guid id);
20         int Delete(T entity);
21         Task<int> DeleteAsync(T entity);
22         T Get(Guid? id);
23         Task<T> GetAsync(Guid? id);
24         List<T> GetAll();
25         Task<List<T>> GetAllAsync();
26
27         List<T> Query(string sql);
28         Task<List<T>> QueryAsync(string sql);
29         List<T> Query(string sql, object[] sqlParametersObject);
30         Task<List<T>> QueryAsync(string sql, object[] sqlParametersObject);
31     }
32 }
33
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Data.Entity;
7  using System.Data.Entity.Infrastructure;
8  using System.Data.Entity.Validation;
9  using C3PO.DAL;
10
11 namespace C3PO.DAL.Lib
12 {
13     public abstract class BaseRepository<T>: IDisposable where T : class, new()
14     {
15         // instantiating the Context object
16         protected readonly DbContext Context;
17         protected DbSet<T> Table;
18
19         public BaseRepository(DbContext context)
20         {
21             Context = context;
22         }
23
24         // Read methods in CRUD:
25
26         public T Get(Guid? id) => Table.Find(id);
27
28         public async Task<T> GetAsync(Guid? id) => await Table.FindAsync(id);
29
30         public List<T> GetAll() => Table.ToList();
31
32         public Task<List<T>> GetAllAsync() => Table.ToListAsync();
33
34         // Create methods (Add and Save) in CRUD
35
```

```
36          public int Add(T entity)
37          {
38              Table.Add(entity);
39              return SaveChanges();
40          }
41
42          public async Task<int> AddAsync(T entity)
43          {
44              Table.Add(entity);
45              return await SaveChangesAsync();
46          }
47
48          public int AddRange(IList<T> entities)
49          {
50              Table.AddRange(entities);
51              return SaveChanges();
52          }
53          public Task<int> AddRangeAsync(IList<T> entities)
54          {
55              Table.AddRange(entities);
56              return SaveChangesAsync();
57          }
58
59          public int AddRange(IEnumerable<T> entities)
60          {
61              Table.AddRange(entities);
62              return SaveChanges();
63          }
64
65          public Task<int> AddRangeAsync(IEnumerable<T> entities)
66          {
67              Table.AddRange(entities);
68              return SaveChangesAsync();
69          }
70
```

```
71          public int Save(T entity)
72          {
73              Context.Entry(entity).State = EntityState.Modified;
74              return SaveChanges();
75          }
76
77          public async Task<int> SaveAsync(T entity)
78          {
79              Context.Entry(entity).State = EntityState.Modified;
80              return await SaveChangesAsync();
81          }
82
83          // Delete methods in CRUD
84
85          public int Delete(T entity)
86          {
87              Context.Entry(entity).State = EntityState.Deleted;
88              return SaveChanges();
89          }
90
91          public async Task<int> DeleteAsync(T entity)
92          {
93              Context.Entry(entity).State = EntityState.Deleted;
94              return await SaveChangesAsync();
95          }
96
97
98          // SQL kørsler mod databasen
99          // NB! injection-attacs
100
101         public List<T> Query(string sql) => Table.SqlQuery(sql).ToList();
102
103         public Task<List<T>> QueryAsync(string sql) => Table.SqlQuery(sql).ToListAsync();
104
105         public List<T> Query(string sql, object[] sqlParametersObjects) => Table.SqlQuery(sql, sqlParametersObjects).ToList
```

```
            ();
106
107         public Task<List<T>> QueryAsync(string sql, object[] sqlParametersObjects) => Table.SqlQuery(sql).ToListAsync();
108
109         // SaveChanges() hjælper methode
110         // Fælles fejlhåndtering for alle metoder der skriver til databasen
111         // lægges i en hjælper metode for at genbruge og minimere vedligeholdelse
112
113         internal int SaveChanges()
114         {
115             try
116             {
117                 return Context.SaveChanges();
118             }
119             catch (DbUpdateConcurrencyException ex)
120             {
121                 //Thrown when there is a concurrency error
122                 //If Entries propery is null, no records were modified
123                 //entities in Entries threw error due to timestamp/conncurrency
124                 //for now, just rethrow the exception
125                 throw;
126             }
127             catch (DbUpdateException ex)
128             {
129                 //Thrown when database update fails
130                 //Examine the inner exception(s) for additional
131                 //details and affected objects
132                 //for now, just rethrow the exception
133                 throw;
134             }
135             catch (CommitFailedException ex)
136             {
137                 //handle transaction failures here
138                 //for now, just rethrow the exception
139                 throw;
```

```
140             }
141             catch (DbEntityValidationException ex)
142             {
143                 foreach (var eve in ex.EntityValidationErrors)
144                 {
145                     Console.WriteLine("Entity of type \"{0}\" in state \"{1}\" has the following validation errors:",
146                         eve.Entry.Entity.GetType().Name, eve.Entry.State);
147                     foreach (var ve in eve.ValidationErrors)
148                     {
149                         Console.WriteLine("- Property: \"{0}\", Value: \"{1}\", Error: \"{2}\"",
150                             ve.PropertyName,
151                             eve.Entry.CurrentValues.GetValue<object>(ve.PropertyName),
152                             ve.ErrorMessage);
153                     }
154                 }
155                 throw;
156             }
157             catch (Exception ex)
158             {
159                 //some other exception happened and should be handled
160                 throw;
161             }
162 
163         }
164 
165         internal async Task<int> SaveChangesAsync()
166         {
167             try
168             {
169                 return await Context.SaveChangesAsync();
170             }
171             catch (DbUpdateConcurrencyException ex)
172             {
173                 //Thrown when there is a concurrency error
174                 //for now, just rethrow the exception
```

```
175                 throw;
176             }
177             catch (DbUpdateException ex)
178             {
179                 //Thrown when database update fails
180                 //Examine the inner exception(s) for additional
181                 //details and affected objects
182                 //for now, just rethrow the exception
183                 throw;
184             }
185             catch (CommitFailedException ex)
186             {
187                 //handle transaction failures here
188                 //for now, just rethrow the exception
189                 throw;
190             }
191             catch (DbEntityValidationException ex)
192             {
193                 foreach (var eve in ex.EntityValidationErrors)
194                 {
195                     Console.WriteLine("Entity of type \"{0}\" in state \"{1}\" has the following validation errors:",
196                         eve.Entry.Entity.GetType().Name, eve.Entry.State);
197                     foreach (var ve in eve.ValidationErrors)
198                     {
199                         Console.WriteLine("- Property: \"{0}\", Value: \"{1}\", Error: \"{2}\"",
200                             ve.PropertyName,
201                             eve.Entry.CurrentValues.GetValue<object>(ve.PropertyName),
202                             ve.ErrorMessage);
203                     }
204                 }
205                 throw;
206             }
207             catch (Exception ex)
208             {
209                 //some other exception happened and should be handled
```

```
210                 throw;
211             }
212         }
213 
214 #region "implementering af IDisposable interface'et"
215         // Oprydning af entitetsobjekter så de ikke længere optager ressourcer
216 
217         bool _disposed = false;
218         public void Dispose()
219         {
220             Dispose(true);
221             GC.SuppressFinalize(this);
222         }
223 
224         protected virtual void Dispose(bool disposing)
225         {
226             if (_disposed)
227                 return;
228 
229             if (disposing)
230             {
231                 Context.Dispose();
232                 // Free any managed objects here.
233                 //
234             }
235 
236             // Free any unmanaged objects here.
237             //
238             _disposed = true;
239         }
240 #endregion
241 
242     }
243 }
244 
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using C3PO.DAL.Models;
7  using C3PO.DAL.Lib;
8
9  namespace C3PO.DAL.Repository
10 {
11     public class BusinessModelRepository : BaseRepository<BusinessModel>, IRepository<BusinessModel>
12     {
13         public BusinessModelRepository(C3POContext dataContext) : base(dataContext)
14         {
15             Table = DataContext.eBusinessModel;
16         }
17
18         public C3POContext DataContext
19         {
20             get { return Context as C3POContext; }
21         }
22
23         public int Delete(Guid guid)
24         {
25             DataContext.Entry(new BusinessModel() { ModelId = guid }).State = System.Data.Entity.EntityState.Deleted;
26             return SaveChanges();
27         }
28
29         public async Task<int> DeleteAsync(Guid guid)
30         {
31             DataContext.Entry(new BusinessModel() { ModelId = guid }).State = System.Data.Entity.EntityState.Deleted;
32             return await SaveChangesAsync();
33         }
34     }
35 }
```

```
1  using System;
2  using System.Data.Entity;
3  using System.Linq;
4  using C3PO.DAL.Models;
5
6  namespace C3PO.DAL
7  {
8      public class C3POContext : DbContext
9      {
10
11         // If you wish to target a different database and/or database provider, modify the 'C3POEntities'
12         // connection string in the application configuration file.
13         public C3POContext() : base("name=C3POConn")
14         {
15             Database.SetInitializer(new DatabaseInitializer());
16         }
17
18         // Add a DbSet for each entity type that you want to include in your model. For more information
19         // on configuring and using a Code First model, see http://go.microsoft.com/fwlink/?LinkId=390109.
20
21         // public virtual DbSet<MyEntity> MyEntities { get; set; }
22         public virtual DbSet<BusinessModel> eBusinessModel { get; set; }
23         public virtual DbSet<Process> eProcess { get; set; }
24         public virtual DbSet<Activity> eActivity { get; set; }
25         public virtual DbSet<Priority> ePriority { get; set; }
26         public virtual DbSet<Activitywork> eActivitywork { get; set; }
27         public virtual DbSet<Activitylink> rActivitylink { get; set; }
28
29         protected override void OnModelCreating(DbModelBuilder modelBuilder)
30         {
31             modelBuilder.Entity<Activitylink>()
32                 .HasOptional<Activity>(l => l.InputActivity)
33                 .WithMany()
34                 .WillCascadeOnDelete(true);
35
```

```
36              modelBuilder.Entity<Activitylink>()
37                  .HasOptional<Activity>(l => l.OutputActivity)
38                  .WithMany()
39                  .WillCascadeOnDelete(false);
40
41
42          }
43      }
44
45
46  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using C3PO.DAL.Repository;
7  using C3PO.DAL.Models;
8  using C3PO.DAL.Lib;
9
10 namespace C3PO.DAL
11 {
12     public sealed class UnitOfWork : IDisposable
13     {
14         private readonly C3POContext _dataContext;
15         private bool disposed;
16
17         public BusinessModelRepository BusinessModelRepo { get; private set; }
18         public ProcessRepository ProcessRepo { get; private set; }
19         public ActivityRepository ActivityRepo { get; private set; }
20         public ActivitylinkRepository ActivitylinkRepo { get; set; }
21         public ActivityworkRepository ActivityworkRepo { get; set; }
22         public PriorityRepository PriorityRepo { get; set; }
23
24         public UnitOfWork()
25         {
26             _dataContext = new C3POContext();
27             BusinessModelRepo = new BusinessModelRepository(DataContext);
28             ProcessRepo = new ProcessRepository(DataContext);
29             ActivityRepo = new ActivityRepository(DataContext);
30             ActivitylinkRepo = new ActivitylinkRepository(DataContext);
31             ActivityworkRepo = new ActivityworkRepository(DataContext);
32             PriorityRepo = new PriorityRepository(DataContext);
33         }
34
35         public C3POContext DataContext
```

```
36          {
37              get { return _dataContext; }
38          }
39
40          public void SaveChanges()
41          {
42              DataContext.SaveChanges();
43          }
44
45          public BusinessModel AddNewBusinessModel(BusinessModel businessModel)
46          {
47              Validate.RequiredString(businessModel.Name);
48              Validate.RequiredString(businessModel.Description);
49              Validate.RequiredString(businessModel.ConnectionString);
50              Validate.RequiredString(businessModel.Sql);
51
52              BusinessModelRepo.Add(businessModel);
53              SaveChanges();
54              return businessModel;
55          }
56
57          public Process AddNewProcessToBusinessModel(Guid modelId, Process process)
58          {
59              Validate.RequiredString(process.Name);
60              Validate.RequiredString(process.Description);
61
62              BusinessModel model = BusinessModelRepo.Get(modelId);
63              model.ProcessList.Add(process);
64              SaveChanges();
65              return process;
66          }
67
68
69      #region "IDisposable"
70
```

```
71          public void Dispose()
72          {
73              Dispose(true);
74              GC.SuppressFinalize(this);
75          }
76
77          private void Dispose(bool disposing)
78          {
79              if (disposed || !disposing) { return; }
80              if (DataContext != null) { DataContext.Dispose(); }
81
82              disposed = true;
83          }
84  #endregion
85
86      }
87  }
88
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Data.Entity;
7  using C3PO.DAL.Models;
8
9  namespace C3PO.DAL
10 {
11     //public class DatabaseInitializer : DropCreateDatabaseIfModelChanges<C3POEntities>
12     //public class DatabaseInitializer : DropCreateDatabaseAlways<C3POContext>
13     public class DatabaseInitializer : CreateDatabaseIfNotExists<C3POContext>
14     {
15
16         protected override void Seed(C3POContext context)
17         {
18             BusinessModel model;
19             Process proc;
20             Priority p;
21             Activitywork w;
22             int lidx = 0;
23
24             // prioriteringer
25             List<string> risikoOrder = new List<string> { "Høj", "Mellem", "Lav" };
26             List<string> komOrder = new List<string>{ "Albertslund", "Birkerød", "Farum", "Ballerup" };
27             List<string> gvomrOrder = new List<string> { "Birkerød", "Birkerød", "Farum", "Ballerup" };
28
29
30             // OPRET MODEL TIL FØDEKÆDEN
31             string modelname = "Fødekæden";
32             string modeldesc = "Model for regionernes gennemførelse af jordforureningsopgaven";
33             string modelconn = "Data Source=localhost;Initial Catalog = Informatik; Integrated Security = True; Connect
                 Timeout = 0; Encrypt=False;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
34             string modelsql = "SELECT * FROM JARLokFremdrift;";
```

```
35
36              model = new BusinessModel { Name=modelname, Description=modeldesc, ConnectionString=modelconn, Sql=modelsql};
37              context.eBusinessModel.Add(model);
38              context.SaveChanges();
39
40              // OPRET BASISSCENARIET
41              proc = new Process { Name = "Basisscenariet RH", Description= "Basisscenarie for fødekæde i RH", LastUpdate =
                  DateTime.Now, Model = model, ModelId = model.ModelId };
42              context.eProcess.Add(proc);
43
44              int aidx = 0;
45
46
47              // OPRET LED I FØDEKÆDEN FOR V1
48              Activity v1 = new Activity { Name = "V1", Index = aidx++, Budget = 1000000, Staff = 7, Poolsize = 30,
                  PoolsPrPerson = 3F, Process = proc, ProcessId = proc.ProcessId };
49              context.eActivity.Add(v1);
50
51              // tilføj prioritering til V1
52              int pidx = 0;
53              p = new Priority { Column = "Kommune", Index = pidx++, Order = komOrder, ActivityId = v1.ActivityId, Activity =
                  v1 };
54              context.ePriority.Add(p);
55
56              // tilføj workflow til V1
57              int widx = 0;
58              w = new Activitywork { Index = widx++, Name = "Udbud", PoolExpence = 0.0F, Duration = 2, ActivityId =
                  v1.ActivityId, Activity = v1 };
59              context.eActivitywork.Add(w);
60
61              w = new Activitywork { Index = widx++, Name = "Gennemførelse", PoolExpence = 40000.0F, PoolStaff = 1.5F, Duration
                   = 12, ActivityId = v1.ActivityId, Activity = v1 };
62              context.eActivitywork.Add(w);
63
64              w = new Activitywork { Index = widx++, Name = "Afslutning", PoolExpence = 40000.0F, PoolStaff = 1, Duration = 3,
```

```
                ActivityId = v1.ActivityId, Activity = v1 };
65              context.eActivitywork.Add(w);
66
67
68
69              // OPRET LED I FØDEKÆDEN FOR V2
70              Activity v2 = new Activity { Name = "V2", Index = aidx++, Budget = 5000000, Staff = 10, Poolsize = 15,
                  PoolsPrPerson = 2F, Process = proc, ProcessId = proc.ProcessId };
71              context.eActivity.Add(v2);
72
73              // tilføj prioritering til V2
74              pidx = 0;
75
76              p = new Priority { Column = "Kommune", Index = pidx++, Order = komOrder, ActivityId = v2.ActivityId, Activity =
                  v2 };
77              context.ePriority.Add(p);
78
79              // tilføj workflow til V2
80              widx = 0;
81              w = new Activitywork { Index = widx++, Name = "Udbud", PoolExpence = 0.0F, PoolStaff = 2, Duration = 2,
                  ActivityId = v2.ActivityId, Activity = v2 };
82              context.eActivitywork.Add(w);
83
84              w = new Activitywork { Index = widx++, Name = "Gennemførelse", PoolExpence = 40000.0F, PoolStaff = 1.5F, Duration
                   = 12, ActivityId = v2.ActivityId, Activity = v2 };
85              context.eActivitywork.Add(w);
86
87              w = new Activitywork { Index = widx++, Name = "Afslutning", PoolExpence = 40000.0F, PoolStaff = 1, Duration = 3,
                  ActivityId = v2.ActivityId, Activity = v2 };
88              context.eActivitywork.Add(w);
89
90
91
92              // OPRET LED I FØDEKÆDEN FOR UNDERSØGELSER
93              Activity us = new Activity { Name = "Undersøgelser", Index = aidx++, Budget = 10000000, Staff = 13, Poolsize = 5,
```

```
                PoolsPrPerson = 1F, Process = proc, ProcessId = proc.ProcessId };
94              context.eActivity.Add(us);
95
96              // tilføj prioritering til undersøgelser
97              pidx = 0;
98              p = new Priority { Column = "Gvomr", Index = pidx++, Order = gvomrOrder, Activity = us, ActivityId =
                  us.ActivityId };
99              context.ePriority.Add(p);
100
101             // tilføj workflow til undersøgelser
102             widx = 0;
103             w = new Activitywork { Index = widx++, Name = "Udbud", PoolExpence = 0.0F, PoolStaff = 2, Duration = 2, Activity
                  = us, ActivityId = us.ActivityId };
104             context.eActivitywork.Add(w);
105
106             w = new Activitywork { Index = widx++, Name = "Gennemførelse", PoolExpence = 40000.0F, PoolStaff = 1.5F, Duration
                   = 12, Activity = us, ActivityId = us.ActivityId };
107             context.eActivitywork.Add(w);
108
109             w = new Activitywork { Index = widx++, Name = "Afslutning", PoolExpence = 40000.0F, PoolStaff = 1, Duration = 3,
                  Activity = us, ActivityId = us.ActivityId };
110             context.eActivitywork.Add(w);
111
112
113
114             // OPRET LED I FØDEKÆDEN FOR AFVÆRGE
115             Activity av = new Activity { Name = "Afværge", Index = aidx++, Budget = 7000000, Staff = 5, Poolsize = 1,
                  PoolsPrPerson = 3F, Process = proc, ProcessId = proc.ProcessId };
116             context.eActivity.Add(av);
117
118             // tilføj prioritering til afværge
119             pidx = 0;
120             p = new Priority { Column = "Gvomr", Index = pidx++, Order = gvomrOrder, Activity = av, ActivityId =
                  av.ActivityId };
121             context.ePriority.Add(p);
```

```
122
123             // tilføj workflow til afværge
124             widx = 0;
125             w = new Activitywork { Index = widx++, Name = "Udbud", PoolExpence = 0.0F, PoolStaff = 2, Duration = 2, Activity
                  = av, ActivityId = av.ActivityId };
126             context.eActivitywork.Add(w);
127
128             w = new Activitywork { Index = widx++, Name = "Gennemførelse", PoolExpence = 40000.0F, PoolStaff = 1.5F, Duration
                   = 12, Activity = av, ActivityId = av.ActivityId };
129             context.eActivitywork.Add(w);
130
131             w = new Activitywork { Index = widx++, Name = "Afslutning", PoolExpence = 40000.0F, PoolStaff = 1, Duration = 3,
                  Activity = av, ActivityId = av.ActivityId };
132             context.eActivitywork.Add(w);
133
134
135             // OPRET LED I FØDEKÆDEN FOR DRIFT
136             Activity dr = new Activity { Name = "Drift", Index = aidx++, Budget = 20000000, Staff = 5, Poolsize = 1,
                  PoolsPrPerson = 10F, Process = proc, ProcessId = proc.ProcessId };
137             context.eActivity.Add(dr);
138
139             // tilføj prioritering til Drift
140             pidx = 0;
141             p = new Priority { Column = "Gvomr", Index = pidx++, Order = gvomrOrder, Activity = dr, ActivityId =
                  dr.ActivityId };
142             context.ePriority.Add(p);
143
144             // tilføj workflow til Drift
145             widx = 0;
146             w = new Activitywork { Index = widx++, Name = "Drift", PoolExpence = 0.0F, PoolStaff = 2, Duration = 2, Activity
                  = dr, ActivityId = dr.ActivityId };
147             context.eActivitywork.Add(w);
148
149
150             // OPRET LED I FØDEKÆDEN FOR MONITERING
```

```
151             Activity mon = new Activity { Name = "Monitering", Index = aidx++, Budget = 3000000, Staff = 0, Poolsize = 1,
                  PoolsPrPerson = 10F, Process = proc, ProcessId = proc.ProcessId };
152             context.eActivity.Add(dr);
153
154             // OPRET LED I FØDEKÆDEN FOR KORTLÆGNING UDEN INDSATS
155             Activity ii = new Activity { Name = "Kortlagt uden indsats", Index = aidx++, Budget = 0, Staff = 0, Poolsize = 1,
                   PoolsPrPerson = 0F, Process = proc, ProcessId = proc.ProcessId };
156             context.eActivity.Add(ii);
157
158
159             // OPRET LED I FØDEKÆDEN FOR Udgået
160             Activity ug = new Activity { Name = "Udgået", Index = aidx++, Budget = 0, Staff = 0, Poolsize = 1, PoolsPrPerson
                  = 0F, Process = proc, ProcessId = proc.ProcessId };
161             context.eActivity.Add(ug);
162
163
164
165             // FORBIND LED I FØDEKØDE MED HITRATER
166             List<Activitylink> links = new List<Activitylink>()
167             {
168                 // V1
169                 new Activitylink { Name = "Høj risiko V1", Index = lidx++, Probability = 0.1F, SiteStatus = "V2", SiteRisk =
                      "Høj", InputActivity = v1, InputActivityId = v1.ActivityId, OutputActivity = v2, OutputActivityId =
                      v2.ActivityId },
170                 new Activitylink { Name = "Mellem risiko V1", Index = lidx++, Probability = 0.1F, SiteStatus = "V2", SiteRisk
                       = "Mellem", InputActivity = v1, InputActivityId = v1.ActivityId, OutputActivity = ii, OutputActivityId =
                      ii.ActivityId },
171                 new Activitylink { Name = "Lav risiko V1", Index = lidx++, Probability = 0.1F, SiteStatus = "V2", SiteRisk =
                      "Lav", InputActivity = v1, InputActivityId = v1.ActivityId, OutputActivity = ii, OutputActivityId =
                      ii.ActivityId },
172                 new Activitylink { Name = "Ingen risiko V1", Index = lidx++, Probability = 0.1F, SiteStatus = "Udgået",
                      SiteRisk = "Ingen", InputActivity = v1, InputActivityId = v1.ActivityId, OutputActivity = ug,
                      OutputActivityId = ug.ActivityId },
173
174                 // V2
```

```
175                 new Activitylink { Name = "Høj risiko V2", Index = lidx++, Probability = 0.1F, SiteStatus = "V2", SiteRisk =
                      "Høj", InputActivity = v2, InputActivityId = v2.ActivityId, OutputActivity = us, OutputActivityId =
                      us.ActivityId },
176                 new Activitylink { Name = "Mellem risiko V2", Index = lidx++, Probability = 0.1F, SiteStatus = "V2", SiteRisk
                       = "Mellem", InputActivity = v2, InputActivityId = v2.ActivityId, OutputActivity = ii, OutputActivityId =
                      ii.ActivityId },
177                 new Activitylink { Name = "Lav risiko V2", Index = lidx++, Probability = 0.1F, SiteStatus = "V2", SiteRisk =
                      "Lav", InputActivity = v2, InputActivityId = v2.ActivityId, OutputActivity = ii, OutputActivityId =
                      ii.ActivityId },
178                 new Activitylink { Name = "Ingen risiko V2", Index = lidx++, Probability = 0.1F, SiteStatus = "Udgået",
                      SiteRisk = "Ingen", InputActivity = v2, InputActivityId = v2.ActivityId, OutputActivity = ug,
                      OutputActivityId = ug.ActivityId },
179
180                 // Undersøgelser
181                 new Activitylink { Name = "Høj risiko Undersøgelser", Index = lidx++, Probability = 0.1F, SiteStatus = "V2",
                      SiteRisk = "Høj", InputActivity = us, InputActivityId = us.ActivityId, OutputActivity = av,
                      OutputActivityId = av.ActivityId },
182                 new Activitylink { Name = "Mellem risiko Undersøgelser", Index = lidx++, Probability = 0.1F, SiteStatus =
                      "V2", SiteRisk = "Mellem", InputActivity = us, InputActivityId = us.ActivityId, OutputActivity = ii,
                      OutputActivityId = ii.ActivityId },
183                 new Activitylink { Name = "Lav risiko Undersøgelser", Index = lidx++, Probability = 0.1F, SiteStatus = "V2",
                      SiteRisk = "Lav", InputActivity = us, InputActivityId = us.ActivityId, OutputActivity = ii,
                      OutputActivityId = ii.ActivityId },
184                 new Activitylink { Name = "Ingen risiko Undersøgelser", Index = lidx++, Probability = 0.1F, SiteStatus =
                      "Udgået", SiteRisk = "Ingen", InputActivity = us, InputActivityId = us.ActivityId, OutputActivity = ug,
                      OutputActivityId = ug.ActivityId },
185
186                 // Afværge
187                 new Activitylink { Name = "Høj risiko Afværge", Index = lidx++, Probability = 0.1F, SiteStatus = "V2",
                      SiteRisk = "Høj", InputActivity = av, InputActivityId = av.ActivityId, OutputActivity = dr,
                      OutputActivityId = dr.ActivityId },
188                 new Activitylink { Name = "Mellem risiko Afværge", Index = lidx++, Probability = 0.1F, SiteStatus = "V2",
                      SiteRisk = "Mellem", InputActivity = v1, InputActivityId = v1.ActivityId, OutputActivity = v2,
                      OutputActivityId = v2.ActivityId },
189                 new Activitylink { Name = "Lav risiko Afværge", Index = lidx++, Probability = 0.1F, SiteStatus = "V2",
```

```
                    SiteRisk = "Lav", InputActivity = v1, InputActivityId = v1.ActivityId, OutputActivity = v2,
                    OutputActivityId = v2.ActivityId },
190                 new Activitylink { Name = "Ingen risiko Afværge", Index = lidx++, Probability = 0.1F, SiteStatus = "Udgået",
                    SiteRisk = "Ingen", InputActivity = v1, InputActivityId = v1.ActivityId, OutputActivity = v2,
                    OutputActivityId = v2.ActivityId }
191
192
193             };
194
195
196             context.rActivitylink.AddRange(links);
197             context.SaveChanges();
198
199             // OPRET TESTSCENARIE TIL FØDEKÆDEM
200             proc = new Process { Name = "Testscenarie", Description = "Testscenarie for fødekæde i RH", LastUpdate =
                  DateTime.Now, Model = model, ModelId = model.ModelId };
201             context.eProcess.Add(proc);
202             context.SaveChanges();
203
204             // OPRET MODEL
205             modelname = "Testmodel";
206             modeldesc = "Model til test af brugerfladen";
207             modelconn = "Data Source=localhost;Initial Catalog = Informatik; Integrated Security = True; Connect Timeout = 0;
                   Encrypt=False;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
208             modelsql = "SELECT * FROM JARLokFremdrift;";
209
210             model = new BusinessModel { Name = modelname, Description = modeldesc, ConnectionString = modelconn, Sql =
                  modelsql };
211             context.eBusinessModel.Add(model);
212             context.SaveChanges();
213
214             // OPRET TESTSCENARIE
215             proc = new Process { Name = "Test scenarie", Description = "Scenarie til test af brugerfladen", LastUpdate =
                  DateTime.Now, Model = model, ModelId = model.ModelId };
216             context.eProcess.Add(proc);
```

```
217             context.SaveChanges();
218
219
220         }
221     }
222 }
223
```

```
1  using System;
2  using System.Runtime.CompilerServices;
3  using System.ComponentModel.DataAnnotations;
4  using System.ComponentModel.DataAnnotations.Schema;
5  using PropertyChanged;
6
7  namespace C3PO.DAL.Lib
8  {
9      using System.ComponentModel;
10
11     public class ObservableObject : INotifyPropertyChanged
12     {
13
14         public event PropertyChangedEventHandler PropertyChanged;
15
16         public void NotifyPropertyChanged([CallerMemberName] string PropertyName = "")
17         {
18             PropertyChangedEventHandler handler = PropertyChanged;
19
20             if (handler != null)
21             {
22                 handler(this, new PropertyChangedEventArgs(PropertyName));
23             }
24         }
25     }
26 }
27
```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace C3PO.DAL.Lib
{
    public static class Validate
    {
        public static void RequiredString(string property)
        {
            if (property==null) { throw new ArgumentNullException(); }
            if (property.Trim().Length == 0) { throw new ArgumentException(); }
        }
    }
}

```

F. Udskrift af koden: C3PO.ViewModel

```
 1
 2  namespace C3PO.DesktopClient.Lib
 3  {
 4      using System;
 5      using System.Collections;
 6      using System.Collections.Generic;
 7      using System.Collections.ObjectModel;
 8      using System.Linq;
 9      using System.ComponentModel;
10      using System.ComponentModel.DataAnnotations;
11      using C3PO.DAL;
12
13      public abstract class BaseViewModel : IDataErrorInfo, INotifyDataErrorInfo, INotifyPropertyChanged
14      {
15          public UnitOfWork UOW { get; private set; }
16
17          public BaseViewModel (UnitOfWork unitOfWork)
18          {
19              UOW = unitOfWork;
20          }
21
22          #region "Property Changed"
23          public event PropertyChangedEventHandler PropertyChanged;
24
25          public bool IsChanged { get; set; } = false;
26          #endregion
27
28          #region "IDataerrorInfo"
29          // IDataErrorInfo
30
31          public string this[string propertyName]
32          {
33              get { return OnValidate(propertyName); }
34          }
35
```

```
36          public string Error
37          {
38              get
39              {
40                  throw new NotSupportedException();
41              }
42          }
43
44          protected virtual string OnValidate(string propertyName)
45          {
46              var context = new ValidationContext(this)
47              {
48                  MemberName = propertyName
49              };
50
51              var results = new Collection<ValidationResult>();
52              var isValid = Validator.TryValidateObject(this, context, results, true);
53
54              if (!isValid)
55              {
56                  ValidationResult result = results.SingleOrDefault(p => p.MemberNames.Any(memberName => memberName ==
                      propertyName));
57                  return result == null ? null : result.ErrorMessage;
58              }
59
60              return null;
61
62          }
63
64          protected string[] GetErrorsFromAnnotations<T>(string propertyName, T value)
65          {
66              var results = new List<ValidationResult>();
67              var vc = new ValidationContext(this, null, null)
68              { MemberName = propertyName };
69              var isValid = Validator.TryValidateProperty(value, vc, results);
```

```
70
71              return (isValid) ? null : Array.ConvertAll(
72                  results.ToArray(), o => o.ErrorMessage);
73          }
74          #endregion
75
76          #region "INotifyDataErrorInfo
77          //INotifyDataErrorInfo
78          protected readonly Dictionary<string, List<string>> Errors =
79              new Dictionary<string, List<string>>();
80
81          protected void ClearErrors(string propertyName = "")
82          {
83              Errors.Remove(propertyName);
84              OnErrorsChanged(propertyName);
85          }
86
87          protected void AddError(string propertyName, string error)
88          {
89              AddErrors(propertyName, new List<string> { error });
90          }
91
92          protected void AddErrors(string propertyName, IList<string> errors)
93          {
94              var changed = false;
95              if (!Errors.ContainsKey(propertyName))
96              {
97                  Errors.Add(propertyName, new List<string>());
98                  changed = true;
99              }
100             foreach (string x in errors)
101             {
102                 if (Errors[propertyName].Contains(x)) return;
103                 Errors[propertyName].Add(x);
104                 changed = true;
```

```
105             };
106             if (changed)
107             {
108                 OnErrorsChanged(propertyName);
109             }
110         }
111         public IEnumerable GetErrors(string propertyName)
112         {
113             if (string.IsNullOrEmpty(propertyName))
114             {
115                 return Errors.Values;
116             }
117             return Errors.ContainsKey(propertyName) ? Errors[propertyName] : null;
118         }
119 
120         public bool HasErrors => Errors.Count != 0;
121 
122         public event EventHandler<DataErrorsChangedEventArgs> ErrorsChanged;
123 
124         protected void OnErrorsChanged(string propertyName)
125         {
126             ErrorsChanged?.Invoke(this, new DataErrorsChangedEventArgs(propertyName));
127         }
128         #endregion
129     }
130 }
131 
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Collections.ObjectModel;
4  using System.ComponentModel.DataAnnotations;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Windows;
9  using System.Windows.Input;
10 using System.Windows.Controls;
11 using C3PO.DAL;
12 using C3PO.DAL.Models;
13 using C3PO.DesktopClient.Lib;
14 using C3PO.DesktopClient.Cmds;
15 using C3PO.DesktopClient.Views;
16
17
18 namespace C3PO.DesktopClient.ViewModels
19 {
20     public class MainWindowViewModel : BaseViewModel
21     {
22         // Commands
23         private RelayCommand _openScenarioCmd = null;
24         public ICommand OpenScenarioCmd => _openScenarioCmd ?? (_openScenarioCmd = new RelayCommand((scenarioId) =>
             OpenScenario((Guid)scenarioId), CanOpenScenario));
25
26         private bool CanOpenScenario(object obj)
27         {
28             return true;
29         }
30
31         //Fields and Properties
32         private UnitOfWork uow;
33         public ObservableCollection<BusinessModel> BusinessModels { get; set; }
34         public BusinessModel SelectedModel { get; set; }
```

```
35          public ObservableCollection<Process> Processes { get; set; }
36          public Process SelectedProcess { get; set; }
37
38          [Required]
39          [StringLength(50, MinimumLength = 2)]
40          public string ModelDescription { get; set; }
41
42
43          // Constructor
44          public MainWindowViewModel(UnitOfWork uow) : base(uow)
45          {
46              BusinessModels = new ObservableCollection<BusinessModel>(UOW.BusinessModelRepo.GetAll());
47              SelectedModel = BusinessModels.First<BusinessModel>();
48              ModelDescription = SelectedModel.Description;
49              Processes = new ObservableCollection<Process>(SelectedModel.ProcessList);
50              if (Processes.Count() > 0)
51              {
52                  SelectedProcess = Processes.First<Process>();
53              }
54          }
55
56          public void OpenScenario(Guid scenarioId)
57          {
58              ProcessWindowViewModel scenariovm = new ProcessWindowViewModel(UOW, scenarioId);
59              var scenarioWindow = new ProcessWindow
60              {
61                  DataContext = scenariovm
62              };
63              scenarioWindow.ShowDialog();
64          }
65
66          // SaveCommand
67          private RelayCommand _saveModelCmd = null;
68          public RelayCommand SaveModelCmd => _saveModelCmd ?? (_saveModelCmd = new RelayCommand(c => SaveModel(), c =>
              CanSaveSelectedModel));
```

```
69
70          public void SaveModel()
71          {
72              SelectedModel.LastUpdate = DateTime.Now;
73              UOW.BusinessModelRepo.Save(SelectedModel);
74              MessageBox.Show("Saved!");
75          }
76
77          private bool CanSaveSelectedModel
78          {
79              get
80              {
81                  if (SelectedModel == null) { return false; }
82                  return SelectedModel.IsChanged;
83              }
84          }
85
86
87
88          // CloseCommand
89          public static readonly ICommand CloseCmd = new RelayCommand(o => ((Window)o).Close());
90
91      }
92  }
93
```

```cs
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;
using C3PO.DAL;
using C3PO.DAL.Models;
using C3PO.DesktopClient.Lib;
using C3PO.DesktopClient.Cmds;

namespace C3PO.DesktopClient.ViewModels
{
    class ProcessWindowViewModel : BaseViewModel
    {
        //Fields and Properties
        public Process Scenario { get; set; }
        public ObservableCollection<Activity> Activities { get; set; }
        public Activity SelectedActivity { get; set; }
        public string CDate { get; set; }
        public string UDate { get; set; }

        // Constructor
        public ProcessWindowViewModel(UnitOfWork uow, Guid ProcessId) : base(uow)
        {
            Scenario = UOW.ProcessRepo.Get(ProcessId);
            Activities = new ObservableCollection<Activity>(Scenario.ActivityList);
            if (Activities.Count() > 0)
            {
                SelectedActivity = Activities.First<Activity>();
            }
            Scenario.IsChanged = false;
```

```
36          }
37
38
39          // Commands
40          //private RelayCmd _openActivityCmd = null;
41          //public RelayCmd OpenActivityCmd => _openActivityCmd ?? (_openActivityCmd = new RelayCmd(c => OpenActivity()));
42          //private RelayCmd _addActivityCmd = null;
43          //public RelayCmd AddActivityCmd => _addActivityCmd ?? (_addActivityCmd = new RelayCmd(c => AddActivity()));
44          //private RelayCmd _deleteActivityCmd = null;
45          //public RelayCmd DeleteActivityCmd => _deleteActivityCmd ?? (_deleteActivityCmd = new RelayCmd(c => DeleteActivity
              ()));
46          //private RelayCmd _activityMoveUpCmd = null;
47          //public RelayCmd ActivityMoveUpCmd => _activityMoveUpCmd ?? (_activityMoveUpCmd = new RelayCmd(c => ActivityMoveUp
              ()));
48          //private RelayCmd _activityMoveDownCmd = null;
49          //public RelayCmd ActivityMoveDownCmd => _activityMoveDownCmd ?? (_activityMoveDownCmd = new RelayCmd(c =>
              ActivityMoveDown()));
50          //private RelayCmd _runScenarioCmd = null;
51          //public RelayCmd RunScenarioCmd => _runScenarioCmd ?? (_runScenarioCmd = new RelayCmd(c => RunScenario()));
52          //private RelayCmd _saveScenarioCmd = null;
53          //public RelayCmd SaveScenarioCmd => _saveScenarioCmd ?? (_saveScenarioCmd = new RelayCmd(c => SaveScenario(), c =>
              CanSaveScenario));
54          //public static readonly ICommand CloseCmd = new RelayCmd(o => ((Window)o).Close());
55
56
57          public void OpenActivity()
58          {
59              MessageBox.Show("Not implemented yet!");
60          }
61
62          public void AddActivity()
63          {
64              MessageBox.Show("Not implemented yet!");
65          }
66
```

```
67          public void DeleteActivity()
68          {
69              MessageBox.Show("Not implemented yet!");
70          }
71
72          public void ActivityMoveUp()
73          {
74              MessageBox.Show("Not implemented yet!");
75          }
76
77          public void ActivityMoveDown()
78          {
79              MessageBox.Show("Not implemented yet!");
80          }
81
82          public void RunScenario()
83          {
84              MessageBox.Show("Not implemented yet!");
85          }
86
87
88          // SaveCommand
89          private RelayCommand _saveScenarioCmd = null;
90          public RelayCommand SaveScenarioCmd => _saveScenarioCmd ?? (_saveScenarioCmd = new RelayCommand(c => SaveScenario(),
              c => CanSaveScenario));
91
92          public void SaveScenario()
93          {
94              Scenario.LastUpdate = DateTime.Now;
95              UOW.ProcessRepo.Save(Scenario);
96          }
97
98          private bool CanSaveScenario
99          {
100             get
```

```
101             {
102                 return Scenario.IsChanged;
103             }
104         }
105 
106     }
107 }
108 
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace C3PO.DesktopClient.Lib
8  {
9      public static class ValidateProperty
10     {
11         public static void RequiredString(string property)
12         {
13             if (property==null) { throw new ArgumentNullException(); }
14             if (property.Trim().Length == 0) { throw new ArgumentException(); }
15         }
16     }
17 }
18
```

```
1  using System;
2  using System.Windows.Input;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace C3PO.DesktopClient.Cmds
9  {
10     public abstract class CmdBase
11     {
12
13         public abstract void Execute(object parameter);
14         public abstract bool CanExecute(object parameter);
15
16         public event EventHandler CanExecuteChanged
17         {
18             add { CommandManager.RequerySuggested += value; }
19             remove { CommandManager.RequerySuggested -= value; }
20         }
21     }
22 }
23
```

```
1  using System;
2  using System.Windows.Input;
3
4  namespace C3PO.DesktopClient.Cmds
5  {
6      public class RelayCmd<T>
7      {
8          private readonly Action<T> _action;
9          private readonly Func<T, bool> _canExecute;
10         public RelayCmd(Action<T> action) : this(action, null) { }
11
12         public RelayCmd(Action<T> action, Func<T, bool> canExecute)
13         {
14             if (action == null) { throw new ArgumentNullException(nameof(action), "Action<T> er et nødvendigt argument"); }
15             _action = action;
16             _canExecute = canExecute;
17         }
18         public bool CanExecute(object parameter) => _canExecute == null || _canExecute((T)parameter);
19         public void Execute(object parameter) { _action((T)parameter); }
20
21         public event EventHandler CanExecuteChanged
22         {
23             add { CommandManager.RequerySuggested += value; }
24             remove { CommandManager.RequerySuggested -= value; }
25         }
26     }
27 }
28
```

```
1  using System;
2  using System.Windows.Input;
3
4  namespace C3PO.DesktopClient.Cmds
5  {
6      public class RelayCommand : ICommand
7      {
8          private Action<object> execute;
9
10         private Predicate<object> canExecute;
11
12         private event EventHandler CanExecuteChangedInternal;
13
14         public RelayCommand(Action<object> execute)
15             : this(execute, DefaultCanExecute)
16         {
17         }
18
19         public RelayCommand(Action<object> execute, Predicate<object> canExecute)
20         {
21             if (execute == null)
22             {
23                 throw new ArgumentNullException("execute");
24             }
25
26             if (canExecute == null)
27             {
28                 throw new ArgumentNullException("canExecute");
29             }
30
31             this.execute = execute;
32             this.canExecute = canExecute;
33         }
34
35         public event EventHandler CanExecuteChanged
```

```
36          {
37              add
38              {
39                  CommandManager.RequerySuggested += value;
40                  this.CanExecuteChangedInternal += value;
41              }
42
43              remove
44              {
45                  CommandManager.RequerySuggested -= value;
46                  this.CanExecuteChangedInternal -= value;
47              }
48          }
49
50          public bool CanExecute(object parameter)
51          {
52              return this.canExecute != null && this.canExecute(parameter);
53          }
54
55          public void Execute(object parameter)
56          {
57              this.execute(parameter);
58          }
59
60          public void OnCanExecuteChanged()
61          {
62              EventHandler handler = this.CanExecuteChangedInternal;
63              if (handler != null)
64              {
65                  //DispatcherHelper.BeginInvokeOnUIThread(() => handler.Invoke(this, EventArgs.Empty));
66                  handler.Invoke(this, EventArgs.Empty);
67              }
68          }
69
70          public void Destroy()
```

```
71          {
72              this.canExecute = _ => false;
73              this.execute = _ => { return; };
74          }
75
76          private static bool DefaultCanExecute(object parameter)
77          {
78              return true;
79          }
80      }
81
82  }
83
```

G. Udskrift af koden: C3PO.DesktopClient

```
1  using System;
2  using System.Collections.Generic;
3  using System.Configuration;
4  using System.Data;
5  using System.Linq;
6  using System.Threading.Tasks;
7  using System.Windows;
8  using C3PO.DAL;
9  using C3PO.DesktopClient.Views;
10 using C3PO.DesktopClient.ViewModels;
11
12
13
14 namespace C3PO.DesktopClient
15 {
16     /// <summary>
17     /// Interaction logic for App.xaml
18     /// </summary>
19     public partial class App : Application
20     {
21         UnitOfWork UOW = new UnitOfWork();
22
23         protected override void OnStartup(StartupEventArgs e)
24         {
25             base.OnStartup(e);
26             var mainwindow = new MainWindow
27             {
28                 DataContext = new MainWindowViewModel(UOW)
29             };
30             mainwindow.ShowDialog();
31         }
32
33     }
34 }
35
```

```
<Window x:Class="C3PO.DesktopClient.Views.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:vm="clr-namespace:C3PO.DesktopClient.ViewModels"
        Title="C3PO - Computerbaseret Prognose, Prioritering, Planlægning og Omkostningsanalyse" Height="600" Width="550"
        WindowStartupLocation="CenterScreen">
    <Window.CommandBindings>
        <CommandBinding Command="{x:Static SystemCommands.CloseWindowCommand}"
                        CanExecute="CloseWindow_CanExec"
                        Executed="CloseWindow_Exec" />
    </Window.CommandBindings>
    <Grid>
        <Grid>
            <Label x:Name="lblModels" Content="Model:" HorizontalAlignment="Left" Margin="8,10,0,0" VerticalAlignment="Top"
              Width="57"/>
            <ComboBox x:Name="cboModels" HorizontalAlignment="Left" Height="26" Margin="70,10,0,0" VerticalAlignment="Top"
              Width="448"
                ItemsSource="{Binding Path=BusinessModels}"
                DisplayMemberPath="Name"
                SelectedItem="{Binding Path=SelectedModel, Mode=TwoWay}"
                SelectedIndex="0"/>
            <TextBox x:Name="txtModelName" Text="{Binding SelectedItem.Description, ElementName=cboModels, Mode=TwoWay,
              UpdateSourceTrigger=PropertyChanged}" Margin="10,48,25,450"/>

            <Label x:Name="lblProcesses" Content="Scenarier:" HorizontalAlignment="Left" Margin="10,130,0,0"
              VerticalAlignment="Top" Width="112"/>
            <ListBox x:Name="lstProcesses" Height="354" Width="320" Margin="10,160,0,0" VerticalAlignment="Top"
              HorizontalAlignment="Left" HorizontalContentAlignment="Stretch"
               ItemsSource="{Binding SelectedItem.ProcessList, ElementName=cboModels}"
               SelectedItem="{Binding Path=SelectedProcess, Mode=TwoWay}" SelectedValuePath="ProcessId"
               MouseDoubleClick="lstProcesses_MouseDoubleClick">

```

```
31                 <ListBox.ItemTemplate>
32                     <DataTemplate>
33                         <Grid Margin="0,2">
34                             <Grid.ColumnDefinitions>
35                                 <ColumnDefinition Width="0" />
36                                 <ColumnDefinition Width="*" />
37                                 <ColumnDefinition Width="50" />
38                             </Grid.ColumnDefinitions>
39                             <TextBlock Grid.Column="0" Text="{Binding ProcessId}" />
40                             <TextBlock Grid.Column="1" Text="{Binding Name}" />
41                             <TextBlock Grid.Column="2" Text="{Binding User}" />
42                         </Grid>
43                     </DataTemplate>
44                 </ListBox.ItemTemplate>
45 
46             </ListBox>
47             <StackPanel HorizontalAlignment="Right" VerticalAlignment="Top" Width="130"  Height="100" Margin="0,130,10,10">
48                 <Button x:Name="btnSave" Content="Gem" HorizontalAlignment="Center" Height="31" Margin="5" Width="112"
49                     Command="{Binding Path = DataContext.SaveModelCmd,
50                     RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}"
51                     />
52                 <TextBox x:Name="txtLastUpdate" IsEnabled="False" IsReadOnly="True"
53                          Text="{Binding SelectedItem.LastUpdate, ElementName=cboModels, Mode=OneWay, StringFormat=dd-MM-yyyy
                               HH:mm:ss ,UpdateSourceTrigger=PropertyChanged}"/>
54             </StackPanel>
55             <StackPanel HorizontalAlignment="Right" VerticalAlignment="Bottom" Width="130"  Height="180" Margin="0,0,10,10">
56                 <Button x:Name="btnOpen" Content="Åben Scenarie" HorizontalAlignment="Center" Height="31" Margin="5"
                     Width="112"
57                     />
58 
59                 <Button x:Name="btnAdd" Content="Tilføj Scenarie" HorizontalAlignment="Center" Height="31"  Margin="5"
                     Width="112"
60                     />
61                 <Button x:Name="btnDelete" Content="Slet Scenarie" HorizontalAlignment="Center" Height="31"  Margin="5"
                     Width="112"
```

```
62                     />
63                 <Button x:Name="btnClose" Content="Afslut" HorizontalAlignment="Center" Height="31"  Margin="5" Width="112"
64                         Command="{x:Static SystemCommands.CloseWindowCommand}"/>
65             </StackPanel>
66         </Grid>
67     </Grid>
68 </Window>
69
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Windows;
7  using System.Windows.Controls;
8  using System.Windows.Data;
9  using System.Windows.Documents;
10 using System.Windows.Input;
11 using System.Windows.Media;
12 using System.Windows.Media.Imaging;
13 using System.Windows.Navigation;
14 using System.Windows.Shapes;
15 using C3PO.DAL;
16 using C3PO.DesktopClient.ViewModels;
17
18 namespace C3PO.DesktopClient.Views
19 {
20     /// <summary>
21     /// Interaction logic for MainWindow.xaml
22     /// </summary>
23     public partial class MainWindow : Window
24     {
25         public MainWindow()
26         {
27             InitializeComponent();
28         }
29
30         private void CloseWindow_CanExec(object sender, CanExecuteRoutedEventArgs e)
31         {
32             e.CanExecute = true;
33         }
34
35         private void CloseWindow_Exec(object sender, ExecutedRoutedEventArgs e)
```

```
36          {
37              SystemCommands.CloseWindow(this);
38          }
39
40          private void lstProcesses_MouseDoubleClick(object sender, MouseButtonEventArgs e)
41          {
42              FrameworkElement fe = sender as FrameworkElement;
43              ((MainWindowViewModel)fe.DataContext).OpenScenario((Guid)lstProcesses.SelectedValue);
44          }
45      }
46  }
47
```

```
<Window x:Class="C3PO.DesktopClient.Views.ProcessWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:C3PO.DesktopClient"
        mc:Ignorable="d"
        Title="Modelscenarie" Height="610" Width="420"
        WindowStartupLocation="CenterScreen">
    <Window.Resources>
        <Style TargetType="{x:Type TextBox}">
            <Style.Triggers>
                <Trigger Property="Validation.HasError" Value="true">
                    <Setter Property="Background" Value="Pink" />
                    <Setter Property="Foreground" Value="Black" />
                    <Setter Property="ToolTip"
                        Value="{Binding RelativeSource={RelativeSource Self},
                        Path=(Validation.Errors)[0].ErrorContent}"/>
                </Trigger>
            </Style.Triggers>
            <Setter Property="Validation.ErrorTemplate">
                <Setter.Value>
                    <ControlTemplate>
                        <DockPanel LastChildFill="True">
                            <TextBlock Foreground="Red" FontSize="20" Text="!"
                                ToolTip="{Binding ElementName=controlWithError,
                                Path=AdornedElement.(Validation.Errors)[0].ErrorContent}"/>
                            <Border BorderBrush="Red" BorderThickness="1">
                                <AdornedElementPlaceholder Name="controlWithError" />
                            </Border>
                        </DockPanel>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
```

```
36      </Window.Resources>
37      <Window.CommandBindings>
38          <CommandBinding Command="{x:Static SystemCommands.CloseWindowCommand}"
39                          CanExecute="CloseWindow_CanExec"
40                          Executed="CloseWindow_Exec" />
41      </Window.CommandBindings>
42      <Grid>
43          <Grid.RowDefinitions>
44              <RowDefinition Height="180"/>
45              <RowDefinition Height="350"/>
46              <RowDefinition Height="Auto"/>
47          </Grid.RowDefinitions>
48
49          <Grid Grid.Row="0">
50
51              <Grid.ColumnDefinitions>
52                  <ColumnDefinition Width="Auto" SharedSizeGroup="Labels"/>
53                  <ColumnDefinition Width="*"/>
54              </Grid.ColumnDefinitions>
55              <StackPanel Grid.Column="0" Width="100">
56                  <StackPanel.Resources>
57                      <Style TargetType="{x:Type Label}">
58                          <Setter Property="Height" Value="29"/>
59                      </Style>
60                  </StackPanel.Resources>
61                  <Label x:Name="lblName" Content="Navn"/>
62                  <Label x:Name="lblDescription" Content="Beskrivelse" Height="50"/>
63                  <Label x:Name="lblUser" Content="Ejer"/>
64                  <Label x:Name="lblcreated" Content="Oprettet"/>
65                  <Label x:Name="lblUpdate" Content="Sidst opdateret"/>
66              </StackPanel>
67              <StackPanel Grid.Column="1" Width="300">
68                  <StackPanel.Resources>
69                      <Style TargetType="{x:Type TextBox}">
70                          <Setter Property="Height" Value="23"/>
```

```
71                          <Setter Property="Margin" Value="2"/>
72                      </Style>
73                  </StackPanel.Resources>
74                  <TextBox x:Name="txtName"
75                           Text="{Binding Path=Scenario.Name, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>
76                  <TextBox x:Name="txtDescription" TextWrapping="Wrap" Height="50"
77                           Text="{Binding Path=Scenario.Description, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>
78                  <TextBox x:Name="txtUser" Background="AntiqueWhite" IsEnabled="False" IsReadOnly="True"
79                           Text="{Binding Path=Scenario.User, Mode=OneWay, UpdateSourceTrigger=PropertyChanged}" />
80                  <TextBox x:Name="txtcreated" Text="{Binding Path=Scenario.CreatedDate, Mode=OneWay, StringFormat=dd-MM-yyyy
                      HH:mm:ss ,UpdateSourceTrigger=PropertyChanged}"
81                           Background="AntiqueWhite" IsEnabled="False" IsReadOnly="True"/>
82                  <TextBox x:Name="txtUpdate" Background="AntiqueWhite" IsEnabled="False" IsReadOnly="True"
83                           Text="{Binding Path=Scenario.LastUpdate, Mode=OneWay, StringFormat=dd-MM-yyyy
                              HH:mm:ss ,UpdateSourceTrigger=PropertyChanged}"/>
84              </StackPanel>
85              <Separator VerticalAlignment="Bottom" Grid.ColumnSpan="2"/>
86          </Grid>
87      <Grid Grid.Row="1" Margin="0,10,0,0">
88              <Grid.ColumnDefinitions>
89                  <ColumnDefinition Width="Auto" SharedSizeGroup="ProcessList"/>
90                  <ColumnDefinition Width="*"/>
91              </Grid.ColumnDefinitions>
92              <Label Grid.Column="0" Content="Aktiviteter i proces:" Margin="10,0,0,0"/>
93              <ListBox x:Name="lstActivities" Grid.Column="0" Height="300" Margin="10,29,0,0" VerticalAlignment="Top"
                  Width="255"
94                    ItemsSource="{Binding Path=Scenario.ActivityList}"
95                    SelectedItem="{Binding Path=SelectedActivity, Mode=TwoWay}" SelectedValuePath="ActivityId"
96                    MouseDoubleClick="ListBox_MouseDoubleClick"
97                    >
98                  <ListBox.ItemTemplate>
99                      <DataTemplate>
100                         <Grid Margin="0,2">
101                             <Grid.ColumnDefinitions>
102                                 <ColumnDefinition Width="0" />
```

```
103                                 <ColumnDefinition Width="*" />
104                             </Grid.ColumnDefinitions>
105                             <TextBlock Grid.Column="0" Text="{Binding ActivityId}" />
106                             <TextBlock Grid.Column="1" Text="{Binding Name}" />
107                         </Grid>
108                     </DataTemplate>
109                 </ListBox.ItemTemplate>
110 
111             </ListBox>
112             <StackPanel Grid.Column="1" VerticalAlignment="Top" Orientation="Vertical" Margin="15,29,0,10">
113                 <StackPanel.Resources>
114                     <Style TargetType="{x:Type Button}">
115                         <Setter Property="Margin" Value="0,5,0,0"/>
116                         <Setter Property="Height" Value="23"/>
117                         <Setter Property="Padding" Value="5,2"/>
118                         <Setter Property="HorizontalAlignment" Value="Left"/>
119                     </Style>
120                 </StackPanel.Resources>
121                 <Button x:Name="btnAdd" Content="_Tilføj Aktivitet" Width="100"
122                         Command="{Binding Path = DataContext.AddActivityCmd,
123                         RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}"/>
124                 <Button x:Name="btnEdit" Content="_Ret Aktivitet" Width="100"
125                         Command="{Binding Path = DataContext.OpenActivityCmd,
126                         RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}"/>
127                 <Button x:Name="btnDelete" Content="_Slet Aktivitet" Width="100"
128                         Command="{Binding Path = DataContext.DeleteActivityCmd,
129                         RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}"/>
130                 <Label Content="Sortering:" Margin="0,30,0,0"/>
131                 <Button x:Name="btmMoveUp" Command="{Binding Path = DataContext.ActivityMoveUpCmd,
132                         RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}">
133                     <Image Source="C:\Users\akoe0033\Documents\Visual Studio 2017\Projects\C3PO\Images\btnArrowUp.png"/>
134                 </Button>
135                 <Button x:Name="btmMoveDown"  Command="{Binding Path = DataContext.ActivityMoveDownCmd,
136                         RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}">
137                     <Image Source="C:\Users\akoe0033\Documents\Visual Studio 2017\Projects\C3PO\Images\btnArrowDown.png"/>
```

```
138                 </Button>
139             </StackPanel>
140         </Grid>
141         <Grid Grid.Row="2">
142             <Separator VerticalAlignment="Top"/>
143             <StackPanel Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2" HorizontalAlignment="Right" Orientation="Horizontal"
                    Margin="0,5,0,5">
144                 <StackPanel.Resources>
145                     <Style TargetType="{x:Type Button}">
146                         <Setter Property="Height" Value="23"/>
147                         <Setter Property="Width" Value="Auto"/>
148                         <Setter Property="Margin" Value="5,5,5,5"/>
149                         <Setter Property="Padding" Value="10,2"/>
150                         <Setter Property="VerticalAlignment" Value="Bottom"/>
151                     </Style>
152                 </StackPanel.Resources>
153                 <Button x:Name="btnRun" Content="_Kør scenariet" Command="{Binding Path = DataContext.RunScenarioCmd,
154                 RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}"/>
155                 <Button x:Name="btnSave" Content="_Gem"
156                     Command="{Binding Path = DataContext.SaveScenarioCmd,
157                     RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}"/>
158                 <Button x:Name="btnClose" Content="_Luk" Command="{x:Static SystemCommands.CloseWindowCommand}"/>
159             </StackPanel>
160         </Grid>
161     </Grid>
162 </Window>
163
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Windows;
7  using System.Windows.Controls;
8  using System.Windows.Data;
9  using System.Windows.Documents;
10 using System.Windows.Input;
11 using System.Windows.Media;
12 using System.Windows.Media.Imaging;
13 using System.Windows.Navigation;
14 using System.Windows.Shapes;
15 using C3PO.DesktopClient.ViewModels;
16
17
18
19 namespace C3PO.DesktopClient.Views
20 {
21     /// <summary>
22     /// Interaction logic for ScenarioWindow.xaml
23     /// </summary>
24     public partial class ProcessWindow : Window
25     {
26         public ProcessWindow()
27         {
28             InitializeComponent();
29         }
30
31         private void CloseWindow_CanExec(object sender, CanExecuteRoutedEventArgs e)
32         {
33             e.CanExecute = true;
34         }
35
```

```
36          private void CloseWindow_Exec(object sender, ExecutedRoutedEventArgs e)
37          {
38              SystemCommands.CloseWindow(this);
39          }
40
41          private void ListBox_MouseDoubleClick(object sender, MouseButtonEventArgs e)
42          {
43              MessageBox.Show("Selection: " + lstActivities.SelectedValuePath);
44          }
45      }
46  }
47
```