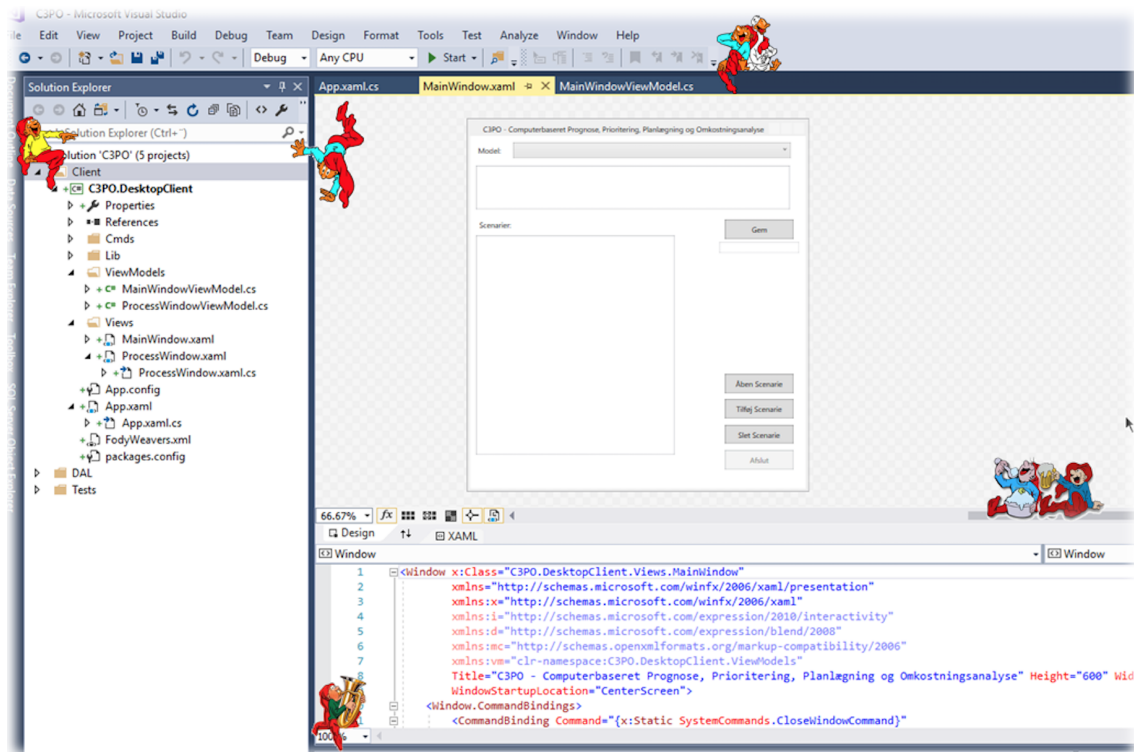




IT DIPLOMUDDANNELSEN

AVANCERET OBJEKTORIENTERET PROGRAMMERING - EFTERÅR 2017



C-3PO: COMPUTERBASERET PROGNOSE, PLANLÆGNING, PRIORITERING OG OMKOSTNINGSANALYSE

STUDERENDE: Allan Kötter
STUDIENUMMER: s140381
UNDERVISER: Henrik Tange
DATO: 19. december 2017
ORDTELLING: 3495 ord

Indhold

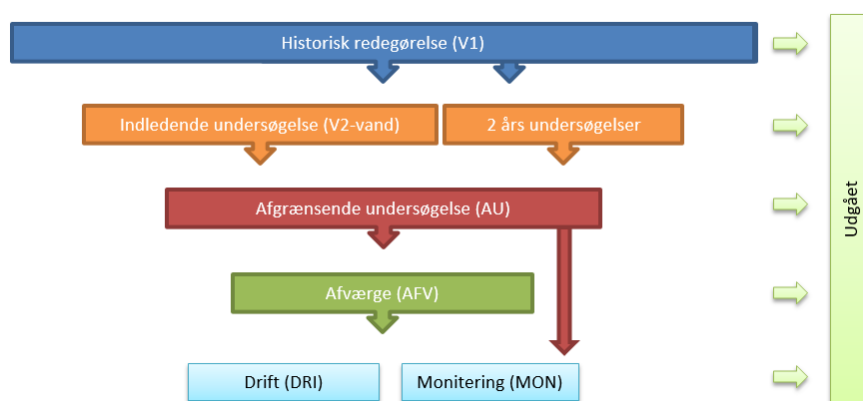
1	Indledning	2
2	Analyse og design	3
2.1	Kravspecifikation	3
2.2	Use Cases	3
2.3	Implementering af forretningslogik	5
2.4	Domænemodel	5
2.5	Overordnet applikations arkitektur	6
3	Bemærkninger til kodens indhold og opbygning	8
3.1	Entitets-klasserne og DbContext	8
3.2	Implementering af <i>Repository</i> mønsteret og UnitOfWork	8
3.3	Implementering af INotifyPropertyChanged	8
3.4	Implementering af interaktiv datavalidering i brugerfladen	9
3.5	Lagdelt arkitektur	9
4	Test	10
4.1	Unittest	10
4.2	Funktionstest	10
5	Selvstudie: Datamodellering med Data Annotations og Fluent API i Entity Framework (EF)	11
5.1	Code First fra et helikopterperspektiv	11
5.2	Attributprogrammering med <i>Data Annotations</i>	12
5.3	The Fluent API	13
5.4	Opsummering	14
6	Konklusion	15
6.1	Resultat	15
6.2	Perspektivering	15
	Referencer	17
	Bilag	18
A	Anvendte værktøjer	19
B	Brugervejledning	20
C	Use Cases	26
D	Klassediagrammer	30

E	Udskrift af koden: C3PO.DAL	32
F	Udskrift af koden: C3PO.ViewModel	76
G	Udskrift af koden: C3PO.DesktopClient	94

1. Indledning

I politisk styrede organisationer, har administrationen ofte behov for at kunne redegøre 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 forøges i hvert 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 bruge applikationen til at opsætte forskellige scenarier med ændrede budgetter og/eller prioriteringsprincipper, for derved bedre at kunne redegøre 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 scenariееjeren, og nye scenarier oprettes med default værdier fra et basisscenarie, og skal derefter kunne ændres af scenariееjeren. Et scenarie skal kunne konfigureres af brugeren som en række sammenkoblede aktiviteter, der hver især omfatter et sæt rammebetingelser herunder tid, økonomi og bemanning 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.

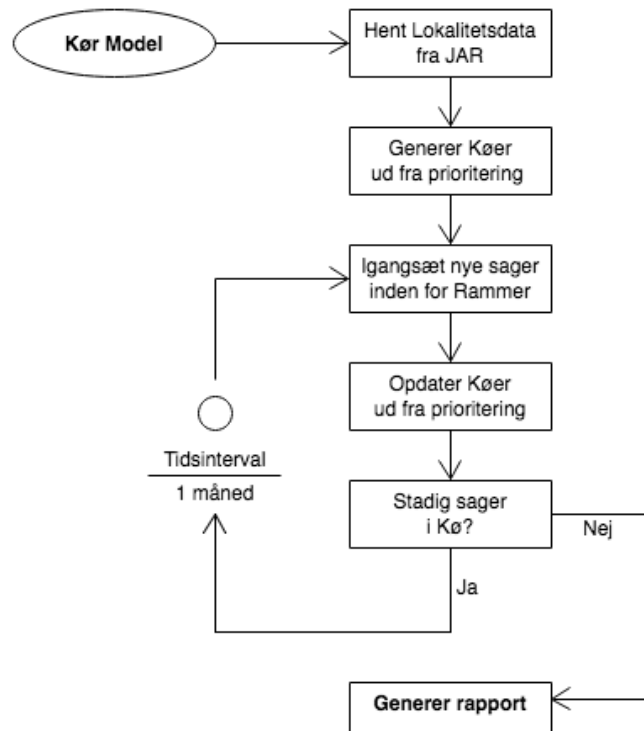
Modelkørslen

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.

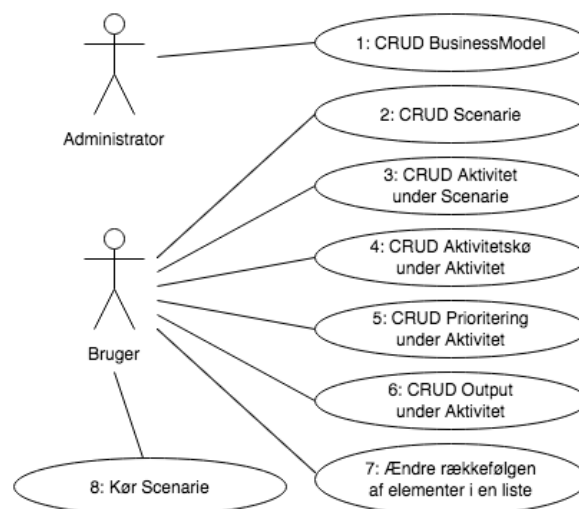
Modelkørslen (figur 2) består af en række iterationer af tidsintervaller på en måned. For hver måned sættes det antal sager i gang på hvert led i fødekæden, som rammebetingelserne tillader herunder budget og personaleressourcer. Når et antal sager sættes i gang, reserveres økonomi og ressourcer til at 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 Usecase-diagram 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 forretningsproces, men være ens for alle scenarier. Der implementeres derfor en datadefinition i forretningsmodellen i form af en connections-tring og en SQL sætning, hvorved et opdateret datasæt kan hentes forud for en modelkørsel.

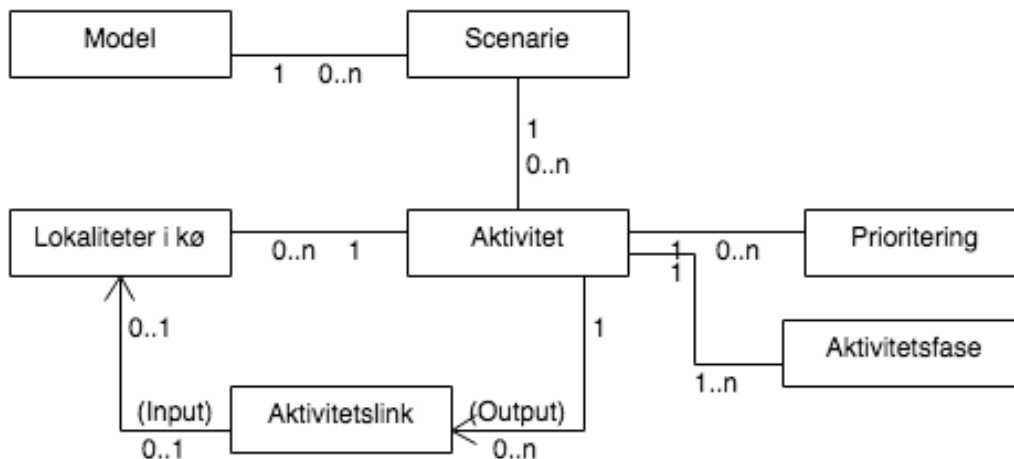
Hver brugerdefineret processcenarie vil indeholde en liste af *aktiviteter*, hvor hvert led i fødekæden kan konfigureres med et sæt rammebetingelser, der bl.a. omfatter budget, personaleressourcer, enhedsomkostninger og sagsbehandlingstid. Disse elementer betragtes som rammebetingelser, fordi de sætter 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 output'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 aktivitet skal være 100%.

2.4 Domænemodel

Domænemodellen er vist på figur 4.



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

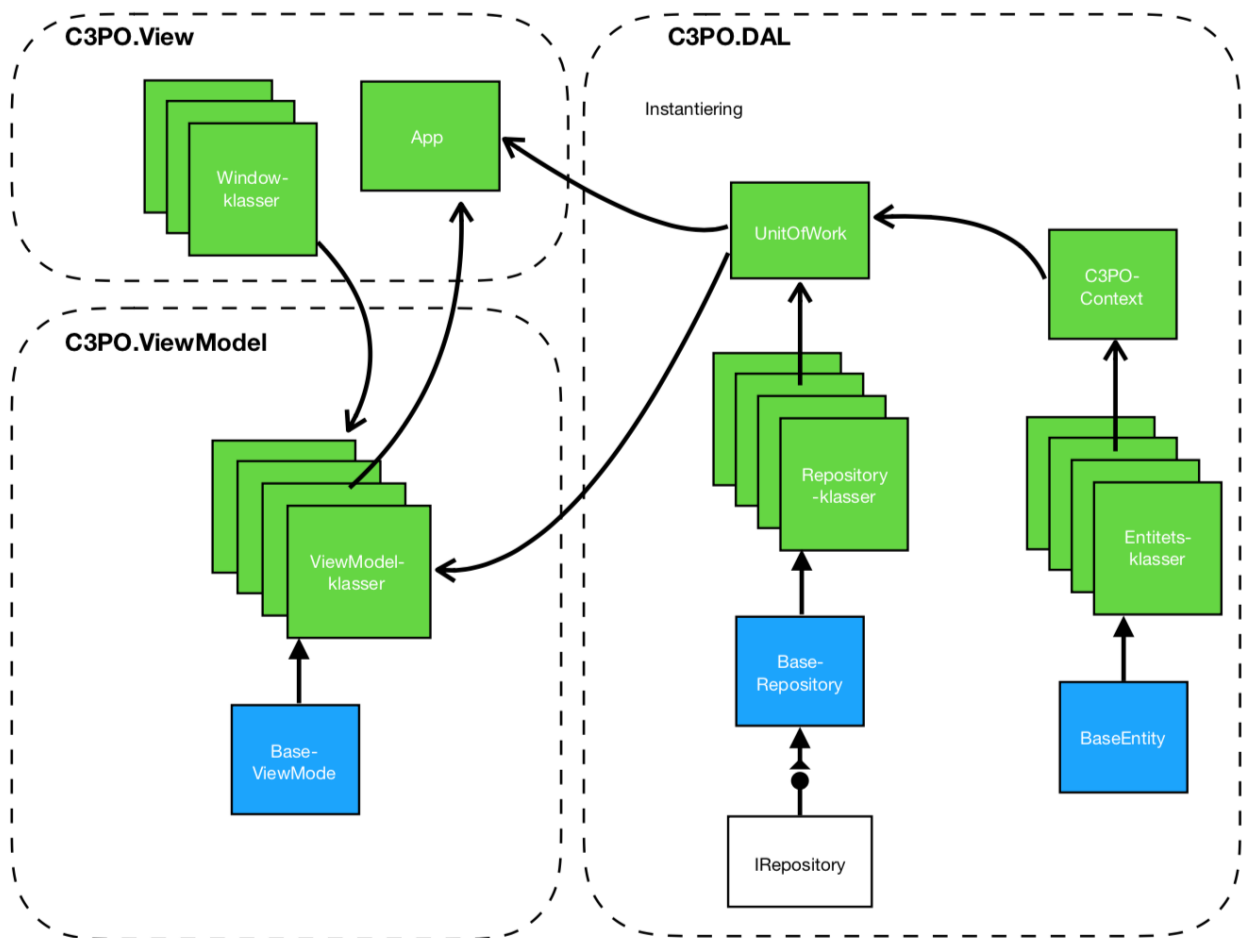
2.5 Overordnet applikations arkitektur

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 separation af data, forretningslogik og præsentation er, uddeling 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 teknolog ved 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.

Selvfølgelig udvikles præsentationslaget vha. WPF-frameworket. For at opnå en klar separation af applikations- og præsentationslaget, implementeres *MVVM*-principperne (Model-View-ViewModel), da dette også er standarden i dag ved udvikling af professionelle WPF-applikationer. Snitfladerne *INotifyPropertyChanged*, *INotifyError* og *INotifyDataErrorInfo* implementeres i forretningslaget og i datalaget, for lettere at opnå en responsiv brugerflade, 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 definerer 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 slettehandlinger.

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 UnitOfWork-klassen, har man automatisk adgang til alle repository-klasserne [6].

3.3 Implementering af INotifyPropertyChanged

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

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 abonnere på ændringer i klassernes properties. Jeg har derfor valgt at installere *PropertyChange.Fody*, og dermed implementere *NotifyPropertyChanged()* 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 generisk 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 samme 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-projekter. 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 demonstrere 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.

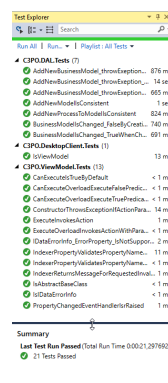
I dette projekt er der oprettet et testprojekt for hver af de 3 projekter C3PO.DAL, C3PO.ViewModel og C3PO.DesktopClient. Testprojekterne er ikke oprettet som en del af applikationens kodestruktur, men er lagt i en separat folder. Kun udvalgte dele af applikationen er testet med TDD, da TDD tager tid og ikke har været 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).

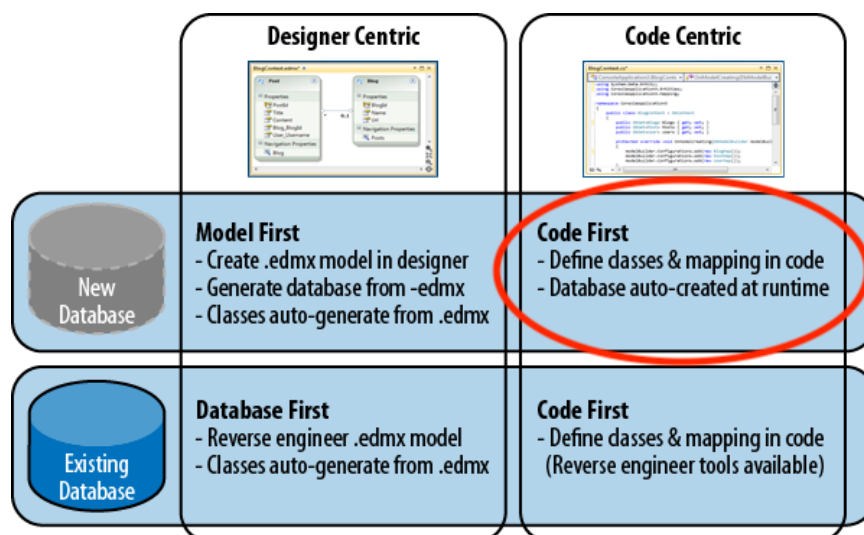


Figur 6: "Screenshot af Visual Studios Test Explorer med gennemførte tests"

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

namespace `System.Data.Entity.ModelConfiguration.Conventions` og som anvendes til en standardiseret oversættelse af EF klassernes `properties` 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 datamodel. 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-generering.

Primærnøgler

Et af de bærende principper 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 classes `property` navngives `Id` eller `<klasse-navn>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 feltpyper 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 standard til databasetypen *nvarchar(max)*. De fleste relationelle databasesystemer 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*-datatypeen 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 konfigurere oversættelsen af en `property` med attributten `[Column]`. `[Column]` 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 standard anvender EF en konvention om at der oprettes "NOT NULL"kolonner for klassernes Id properties og properties med værdi-datatype. Reference-datatyper oprettes tilgængelig alle som "NULL"kolonner. Programmøren kan konfigurere 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 af logikken i datamodellen, kan programmøren vælge at lade databasen generere 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 classes objekter og denne klasse indeholder en property med "forældre objektet", så oversættes dette som en-til-mange relation i databasemodellen, hvorved der oprettes en fremmednøgle i den relaterede tabel-definition. Som standard 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 markere, 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 kontrollere 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 bygger på, og derfor et centralt element i applikationsudvikling. *Data Annotations* giver programmøren mulighed for en stor grad af kontrol over datamodellering, i forbindelse med Entity Framework's *Code First* funktionalitet. Med *The Fluent API* kan programmøren opnå endnu mere finkornet kontrol over datamodellen, og kan konfigurere 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 Framework'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 redegøre 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 fungerende 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](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](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.

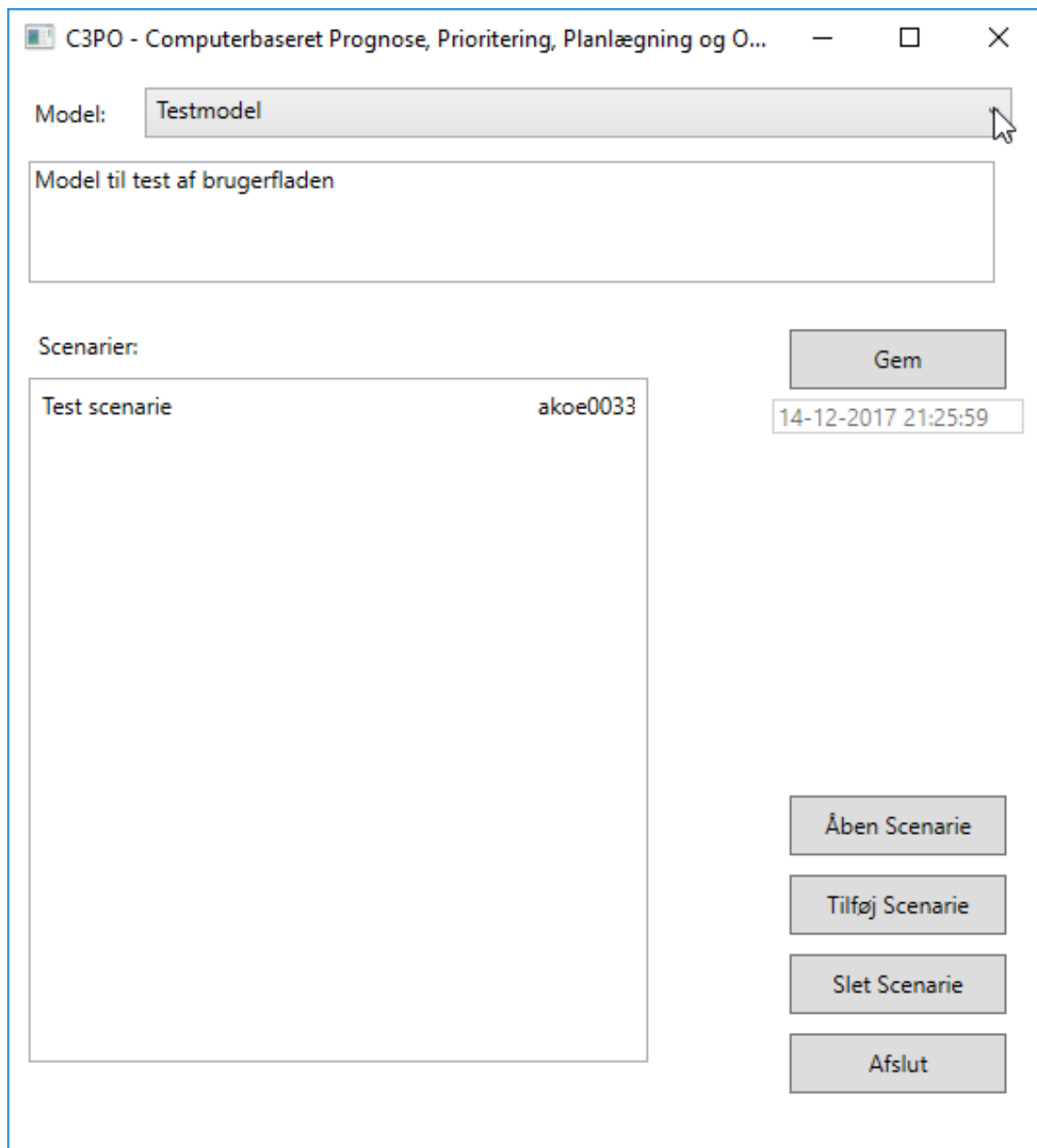
Beskrivelsen kan editeres (figur 9), og ændringerne kan gemmes ved at trykke på knappen "Gem". Tidsfeltet nedenunder knappen afspejler, hvornår seneste ændring er gemt i databasen, og kan benyttes til at verificerer, at ændringerne man har lavet bliver gemt (figur 10). Knapperne "Åben scenarie", "Tilføj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

Scenarie-vinduet

Dobbelt-klikkes på et scenarie i startvinduet's 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

C3PO - Computerbaseret Prognose, Prioritering, Planlægning og O...

Model: Testmodel

Model til test af brugerfladen ... Test

Scenarier:

Test scenarie akoe0033

Gem

14-12-2017 21:25:59

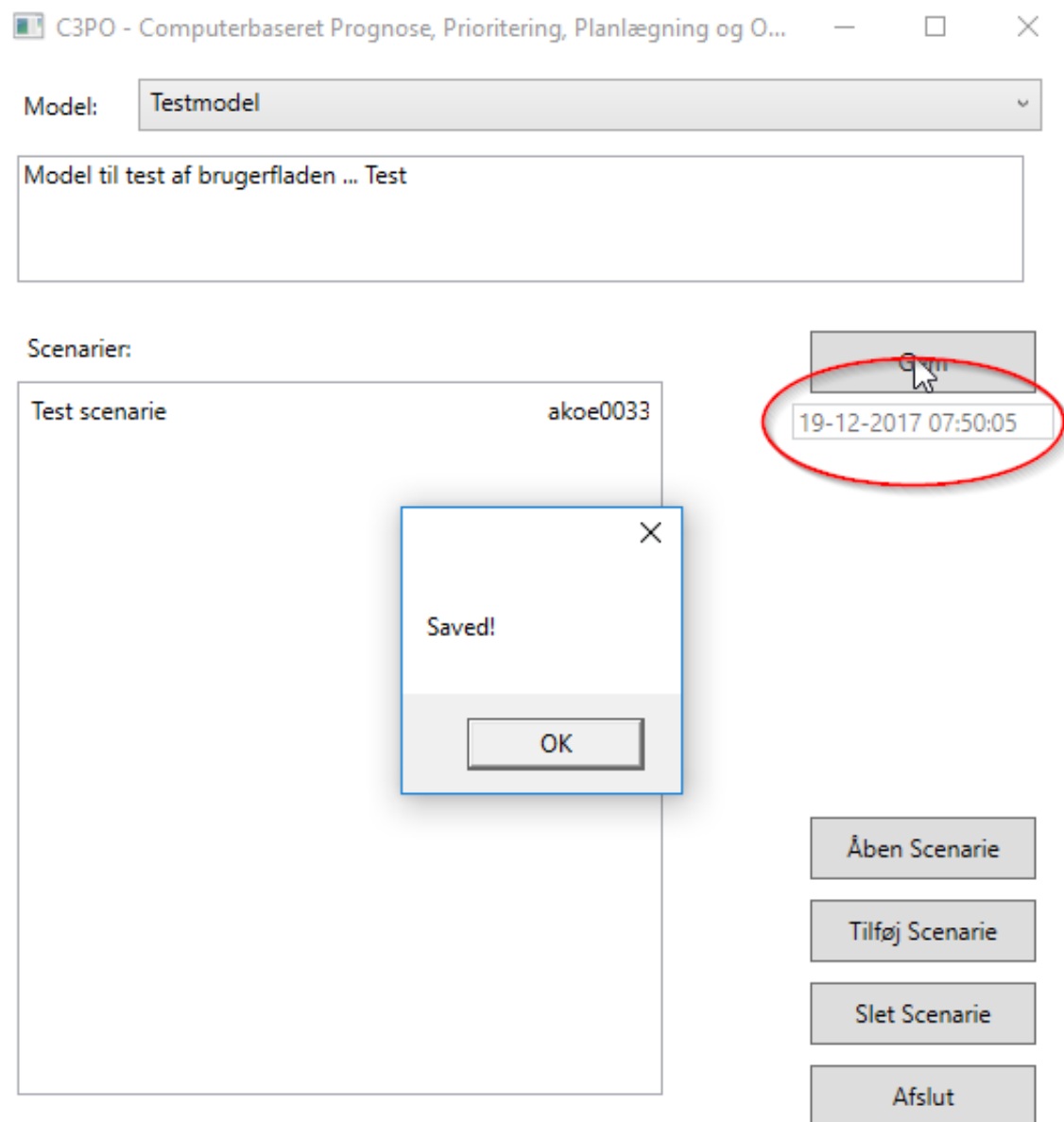
Åben Scenarie

Tilføj Scenarie

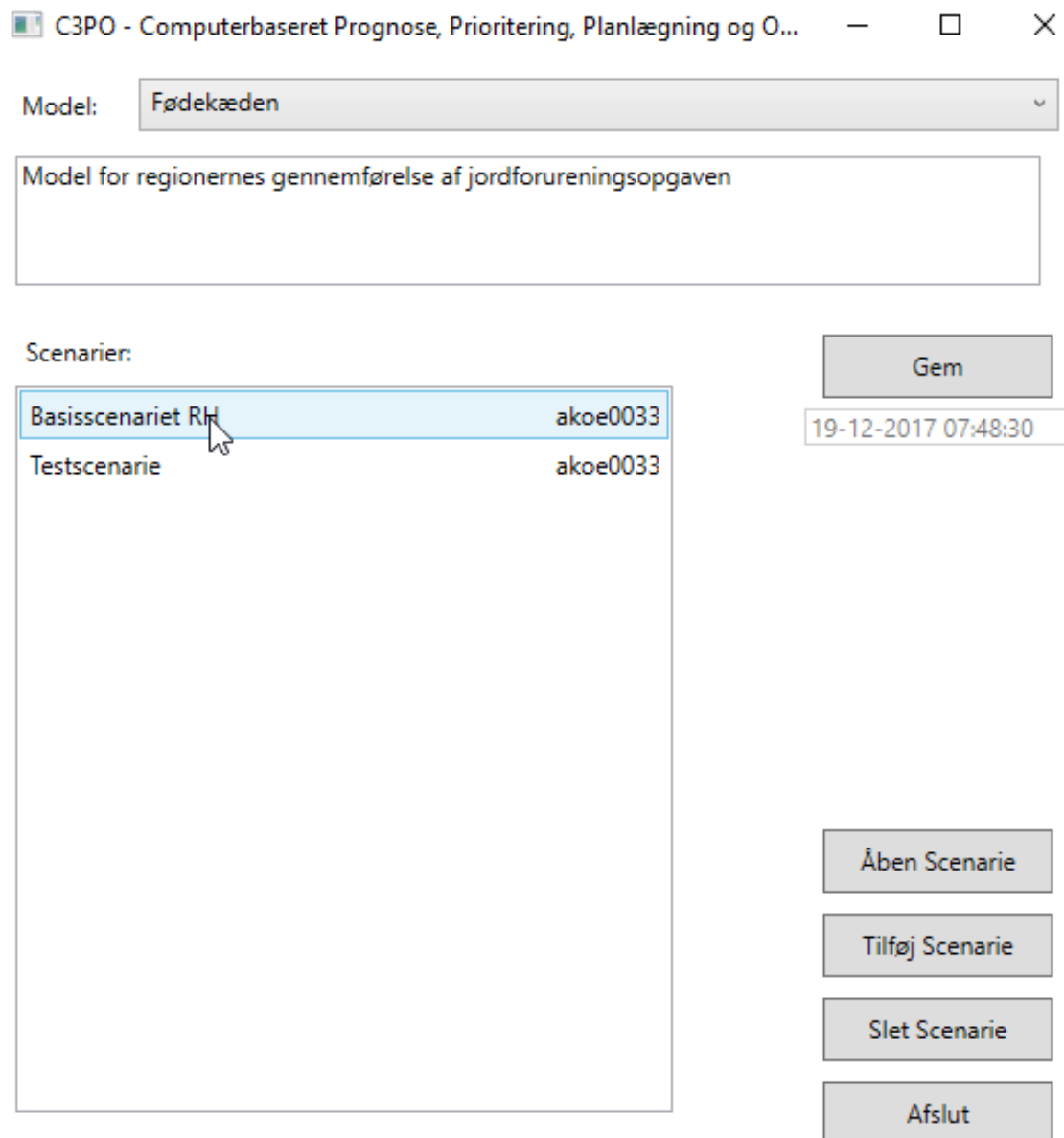
Slet Scenarie

Afslut

Figur 9: Editering af modellens bemærkningsfelt



Figur 10: Ændringer gemmes til databasen



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

Modelscenarie

Navn

Basisscenariet RH

Beskrivelse

Basisscenarie for fødekæde i RH

Ejer

akoe0033

Oprettet

13-12-2017 17:01:19

Sidst opdateret

14-12-2017 22:11:19

Aktiviteter i proces:

V1

V2

Undersøgelser

Afværge

Drift

Kortlagt uden indsats

Udgået

Tilføj Aktivitet

Ret Aktivitet

Slet Aktivitet

Sortering:

▲

▼

Kør scenariet

Gem

Luk

Figur 12: Scenarievinduet, med scenariets data indlæst

25

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 dobbeltklikker 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 oplysningerne i de ønskede felter.
7. Brugeren vælger Gem
8. Modellen gemmes med de ændrede data

Alternativ arbejdsgang 3: Slet

1. Brugeren åbner applikationen og startvindue vises.
2. Brugeren 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 vælger "Slet" for at slette en eksisterende model.
5. Pop-up vindue: "Ønsker du virkelig at slette denne model"
6. Brugeren vælger "Ja"
7. Modeller slettes og fjernes fra listevinduet

UC2: CRUD Scenarie

Aktør: Brugeren

Betingelser: Brugeren har åbnet applikationen 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: Brugeren kan oprette, vise, redigere og slette scenarier i systemet jf CRUD paradigmet.

Primær arbejdsgang: Opret

1. Brugeren 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 Brugers feltet.
4. Brugeren retter evt. ønskede felter.
5. Brugeren vælger Gem
6. Det nye scenarie gemmes

Alternativ arbejdsgang 1: Vis

1. Brugeren dobbeltklikker på et scenarie i listen
2. Et scenarie vindue åbnes og med data for det valgte scenarie.
3. Brugeren 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. Brugeren 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. Brugeren vælger "Ja"
4. Scenariet slettes og fjernes fra listevinduet

UC3: CRUD Aktion

Aktør: Brugeren

Betingelser: Brugeren har åbnet applikationen 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 aktiviteter.

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

Primær arbejdsgang: Opret

1. Brugeren 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 dobbeltklikker på et scenarie i listen
2. Et scenarie vinduet åbnes med data for det valgte scenarie.
3. Brugeren vælger Luk
4. Vinduet lukkes 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. Brugeren 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. Brugeren 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 applikationen 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

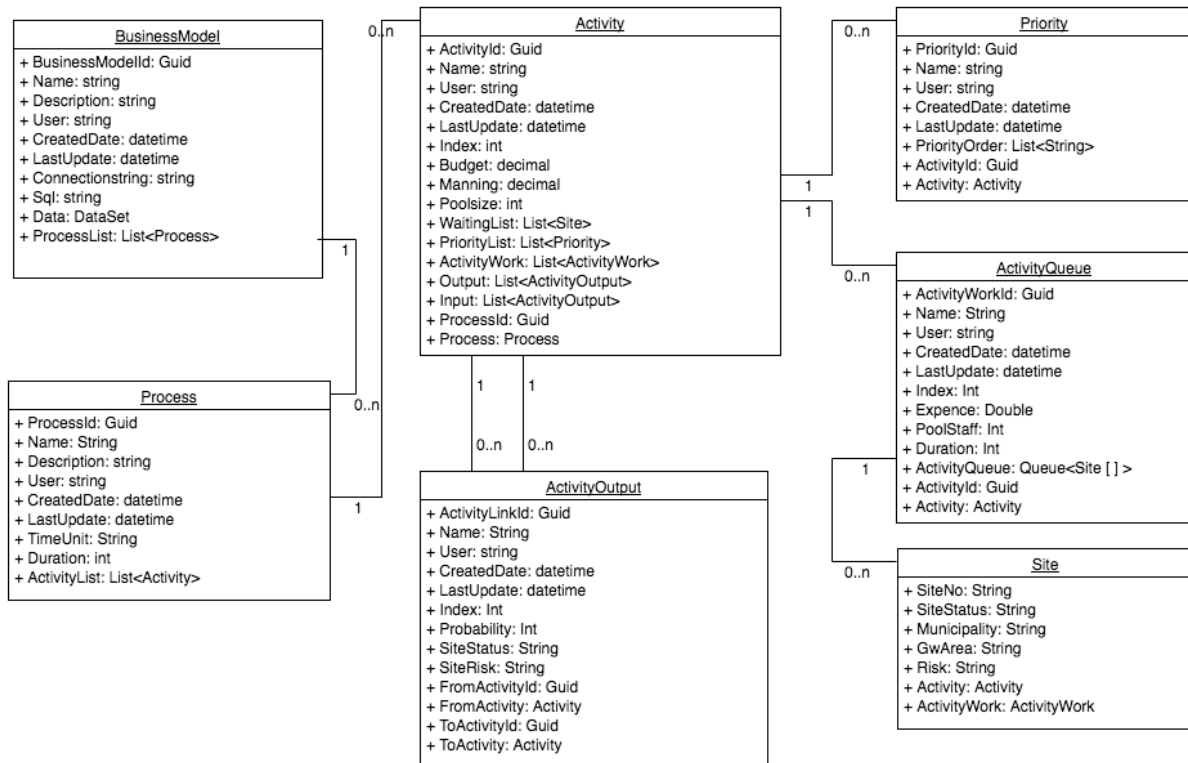
Betingelser: Bruger står i et scenarievindue med et gemt scenarie.

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

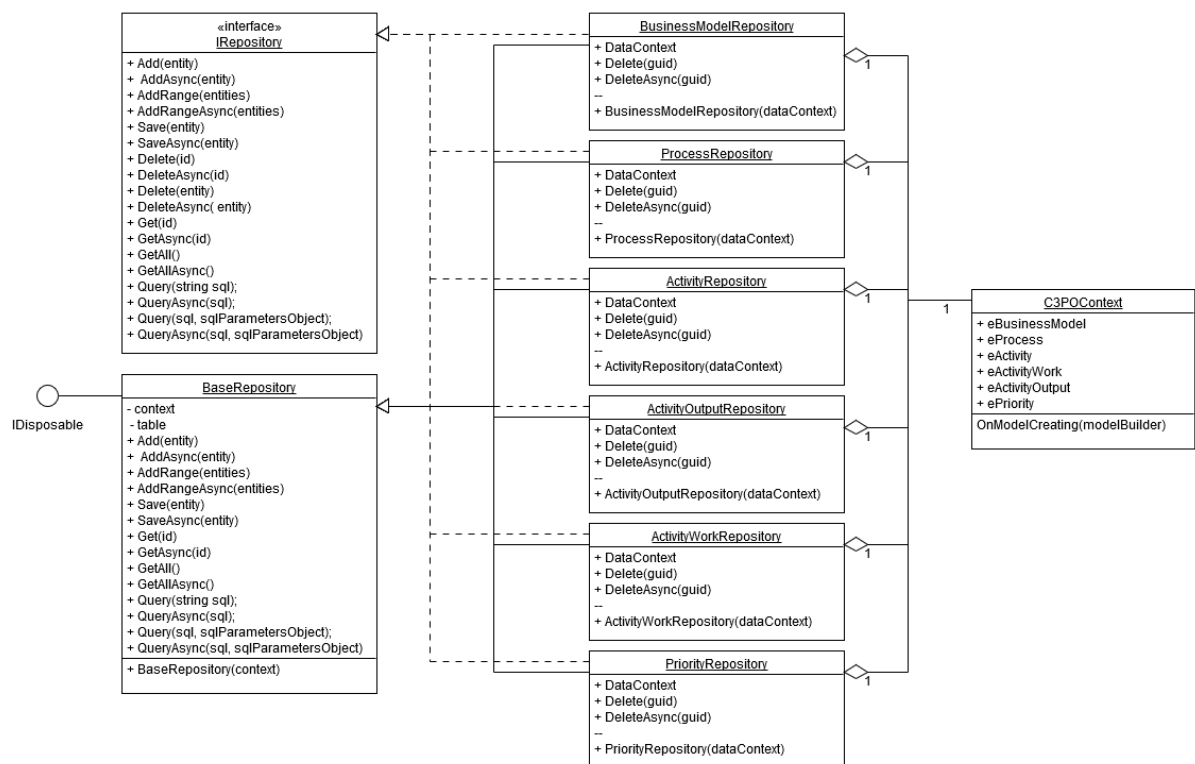
Primær arbejdsgang:

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 ==
65                 propertyName));
66             return result == null ? null : result.ErrorMessage;
67         }
68
69         return null;
```

```
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
```

```
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("eActivitywork")]
15     public partial class Activitywork : BaseModel
16     {
17         // Properties
18         [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
19         public Guid WorkId { get; set; }
20         [Required, StringLength(50)]
21         public string Name { get; set; }
22         [Required]
23         public int Index { get; set; }
24         [Required]
25         public float PoolExpence { get; set; }
26         [Required]
27         public float PoolStaff { get; set; }
28         [Required]
29         public int Duration { get; set; }
30
31         public virtual Queue<Site[]> WQueue { get; set; } = new Queue<Site[]>();
32
33         // Foreign Key property
34         [Required, ForeignKey("Activity"), Column(Order = 0)]
35         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-attacks
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 metode  
110     // Følles 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 property is null, no records were modified  
123             //entities in Entries threw error due to timestamp/concurrency  
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     }  
142 }
```

```
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     }
176 }
```

```
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             // OPRET MODEL TIL FØDEKÆDEN
30             string modelName = "Fødekæden";
31             string modeldesc = "Model for regionernes gennemførelse af jordforureningsopgaven";
32             string modelconn = "Data Source=localhost;Initial Catalog = Informatik; Integrated Security = True; Connect
33                 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", PoolExpen = 0.0F, Duration = 2, ActivityId = ↗
        v1.ActivityId, Activity = v1 };
59     context.eActivitywork.Add(w);
60
61     w = new Activitywork { Index = widx++, Name = "Gennemførelse", PoolExpen = 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", PoolExpen = 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,
71     PoolPrPerson = 2F, Process = proc, ProcessId = proc.ProcessId };
72     context.eActivity.Add(v2);
73
74     // tilføj prioritering til V2
75     pidx = 0;
76
77     p = new Priority { Column = "Kommune", Index = pidx++, Order = komOrder, ActivityId = v2.ActivityId, Activity =
78     v2 };
79     context.ePriority.Add(p);
80
81     // tilføj workflow til V2
82     widx = 0;
83
84     w = new Activitywork { Index = widx++, Name = "Udbud", PoolExpen = 0.0F, PoolStaff = 2, Duration = 2,
85     ActivityId = v2.ActivityId, Activity = v2 };
86     context.eActivitywork.Add(w);
87
88     w = new Activitywork { Index = widx++, Name = "Gennemførelse", PoolExpen = 40000.0F, PoolStaff = 1.5F, Duration
89     = 12, ActivityId = v2.ActivityId, Activity = v2 };
90     context.eActivitywork.Add(w);
91
92     w = new Activitywork { Index = widx++, Name = "Afslutning", PoolExpen = 40000.0F, PoolStaff = 1, Duration = 3,
93     ActivityId = v2.ActivityId, Activity = v2 };
94     context.eActivitywork.Add(w);
95
96
97     // OPRET LED I FØDEKÆDEN FOR UNDERSØGELSER
98     Activity us = new Activity { Name = "Undersøgelser", Index = aidx++, Budget = 10000000, Staff = 13, Poolsize = 5,
99     PoolPrPerson = 2F, Process = proc, ProcessId = proc.ProcessId };
100    context.eActivity.Add(us);
```

```
    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", PoolExpen = 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", PoolExpen = 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", PoolExpen = 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 afværg
124     widx = 0;
125     w = new Activitywork { Index = widx++, Name = "Udbud", PoolExpen = 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", PoolExpen = 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", PoolExpen = 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", PoolExpen = 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 = "Monitoring", 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     // Afværg  
187     new Activitylink { Name = "Høj risiko Afværg", 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 Afværg", 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 Afværg", 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 Afværg", 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 TESTSCENARIOE TIL FØDEKÆDEM
200     proc = new Process { Name = "Testscenario", Description = "Testscenario 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 TESTSCENARIOE
215     proc = new Process { Name = "Test scenario", Description = "Scenario 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
```

```
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.DAL.Lib
8 {
9     public static class Validate
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
```

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     protected string[] GetErrorsFromAnnotations<T>(string propertyName, T value)
64     {
65         var results = new List<ValidationResult>();
66         var vc = new ValidationContext(this, null, null)
67         { MemberName = propertyName };
68         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) =>
25             OpenScenario((Guid)scenarioId), CanOpenScenario));
26
27         private bool CanOpenScenario(object obj)
28         {
29             return true;
30         }
31
32         //Fields and Properties
33         private UnitOfWork uow;
34         public ObservableCollection<BusinessModel> BusinessModels { get; set; }
35         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
```

```
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 C3PO.DAL;
11 using C3PO.DAL.Models;
12 using C3PO.DesktopClient.Lib;
13 using C3PO.DesktopClient.Cmds;
14
15 namespace C3PO.DesktopClient.ViewModels
16 {
17     class ProcessWindowViewModel : BaseViewModel
18     {
19         //Fields and Properties
20         public Process Scenario { get; set; }
21         public ObservableCollection<Activity> Activities { get; set; }
22         public Activity SelectedActivity { get; set; }
23         public string CDate { get; set; }
24         public string UDate { get; set; }
25
26         // Constructor
27         public ProcessWindowViewModel(UnitOfWork uow, Guid ProcessId) : base(uow)
28         {
29             Scenario = UOW.ProcessRepo.Get(ProcessId);
30             Activities = new ObservableCollection<Activity>(Scenario.ActivityList);
31             if (Activities.Count() > 0)
32             {
33                 SelectedActivity = Activities.First<Activity>();
34             }
35             Scenario.IsChanged = false;
```

```
36     }
37
38
39     // Commands
40     //private RelayCommand _openActivityCmd = null;
41     //public RelayCommand OpenActivityCmd => _openActivityCmd ?? (_openActivityCmd = new RelayCommand(c => OpenActivity()));
42     //private RelayCommand _addActivityCmd = null;
43     //public RelayCommand AddActivityCmd => _addActivityCmd ?? (_addActivityCmd = new RelayCommand(c => AddActivity()));
44     //private RelayCommand _deleteActivityCmd = null;
45     //public RelayCommand DeleteActivityCmd => _deleteActivityCmd ?? (_deleteActivityCmd = new RelayCommand(c => DeleteActivity  ➤
46         ());
47     //private RelayCommand _activityMoveUpCmd = null;
48     //public RelayCommand ActivityMoveUpCmd => _activityMoveUpCmd ?? (_activityMoveUpCmd = new RelayCommand(c => ActivityMoveUp  ➤
49         ());
50     //private RelayCommand _activityMoveDownCmd = null;
51     //public RelayCommand ActivityMoveDownCmd => _activityMoveDownCmd ?? (_activityMoveDownCmd = new RelayCommand(c =>  ➤
52         ActivityMoveDown()));
53     //private RelayCommand _runScenarioCmd = null;
54     //public RelayCommand RunScenarioCmd => _runScenarioCmd ?? (_runScenarioCmd = new RelayCommand(c => RunScenario()));
55     //private RelayCommand _saveScenarioCmd = null;
56     //public RelayCommand SaveScenarioCmd => _saveScenarioCmd ?? (_saveScenarioCmd = new RelayCommand(c => SaveScenario(), c =>  ➤
57         CanSaveScenario));
58     //public static readonly ICommand CloseCmd = new RelayCommand(o => ((Window)o).Close());
59
60     public void OpenActivity()
61     {
62         MessageBox.Show("Not implemented yet!");
63     }
64
65     public void AddActivity()
66     {
67         MessageBox.Show("Not implemented yet!");
68     }
```

```
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(),
91     c => CanSaveScenario));
92
93     public void SaveScenario()
94     {
95         Scenario.LastUpdate = DateTime.Now;
96         UOW.ProcessRepo.Save(Scenario);
97     }
98
99     private bool CanSaveScenario
100    {
101        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 C3P0.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 C3P0.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 C3P0.DesktopClient.Cmds
5 {
6     public class RelayCommand<T>
7     {
8         private readonly Action<T> _action;
9         private readonly Func<T, bool> _canExecute;
10        public RelayCommand(Action<T> action) : this(action, null) { }
11
12        public RelayCommand(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
```

```

1 <Window x:Class="C3PO.DesktopClient.Views.MainWindow"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
5     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
7     xmlns:vm="clr-namespace:C3PO.DesktopClient.ViewModels"
8     Title="C3PO - Computerbaseret Prognose, Prioritering, Planlægning og Omkostningsanalyse" Height="600" Width="550"
9     WindowStartupLocation="CenterScreen">
10 <Window.CommandBindings>
11     <CommandBinding Command="{x:Static SystemCommands.CloseWindowCommand}"
12         CanExecute="CloseWindow_CanExec"
13         Executed="CloseWindow_Exec" />
14 </Window.CommandBindings>
15 <Grid>
16     <Grid>
17         <Label x:Name="lblModels" Content="Model:" HorizontalAlignment="Left" Margin="8,10,0,0" VerticalAlignment="Top" Width="57"/>
18         <ComboBox x:Name="cboModels" HorizontalAlignment="Left" Height="26" Margin="70,10,0,0" VerticalAlignment="Top" Width="448"
19             ItemsSource="{Binding Path=BusinessModels}"
20             DisplayMemberPath="Name"
21             SelectedItem="{Binding Path=SelectedModel, Mode=TwoWay}"
22             SelectedIndex="0"/>
23         <TextBox x:Name="txtModelName" Text="{Binding SelectedItem.Description, ElementName=cboModels, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" Margin="10,48,25,450"/>
24
25         <Label x:Name="lblProcesses" Content="Scenarier:" HorizontalAlignment="Left" Margin="10,130,0,0" VerticalAlignment="Top" Width="112"/>
26         <ListBox x:Name="lstProcesses" Height="354" Width="320" Margin="10,160,0,0" VerticalAlignment="Top" HorizontalAlignment="Left" HorizontalContentAlignment="Stretch"
27             ItemsSource="{Binding SelectedItem.ProcessList, ElementName=cboModels}"
28             SelectedItem="{Binding Path=SelectedProcess, Mode=TwoWay}" SelectedValuePath="ProcessId"
29             MouseDoubleClick="lstProcesses_MouseDoubleClick">
30

```

```

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
54             HH:mm:ss ,UpdateSourceTrigger=PropertyChanged}"/>
55     </StackPanel>
56     <StackPanel HorizontalAlignment="Right" VerticalAlignment="Bottom" Width="130" Height="180" Margin="0,0,10,10">
57         <Button x:Name="btnOpen" Content="Åben Scenarie" HorizontalAlignment="Center" Height="31" Margin="5"
58             Width="112"
59             />
60         <Button x:Name="btnAdd" Content="Tilføj Scenarie" HorizontalAlignment="Center" Height="31" Margin="5"
61             Width="112"
62             />
63         <Button x:Name="btnDelete" Content="Slet Scenarie" HorizontalAlignment="Center" Height="31" Margin="5"
64             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
```

```
1 <Window x:Class="C3P0.DesktopClient.Views.ProcessWindow"
2       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4       xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5       xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6       xmlns:local="clr-namespace:C3P0.DesktopClient"
7       mc:Ignorable="d"
8       Title="Modelscenarie" Height="610" Width="420"
9       WindowStartupLocation="CenterScreen">
10     <Window.Resources>
11       <Style TargetType="{x:Type TextBox}">
12         <Style.Triggers>
13           <Trigger Property="Validation.HasError" Value="true">
14             <Setter Property="Background" Value="Pink" />
15             <Setter Property="Foreground" Value="Black" />
16             <Setter Property="ToolTip"
17               Value="{Binding RelativeSource={RelativeSource Self},
18                 Path=(Validation.Errors)[0].ErrorContent}"/>
19           </Trigger>
20         </Style.Triggers>
21         <Setter Property="Validation.ErrorTemplate">
22           <Setter.Value>
23             <ControlTemplate>
24               <DockPanel LastChildFill="True">
25                 <TextBlock Foreground="Red" FontSize="20" Text="!"
26                   ToolTip="{Binding ElementName=controlWithError,
27                     Path=AdornedElement.(Validation.Errors)[0].ErrorContent}"/>
28                 <Border BorderBrush="Red" BorderThickness="1">
29                   <AdornedElementPlaceholder Name="controlWithError" />
30                 </Border>
31               </DockPanel>
32             </ControlTemplate>
33           </Setter.Value>
34         </Setter>
35       </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"/>

```



```

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="btnMoveUp" Command="{Binding Path = DataContext.ActivityMoveUpCmd,
132         RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}">
133         <Image Source="C:\Users\ * * * * \Documents\Visual Studio 2017\Projects\C3P0\Images\btnArrowUp.png" />
134     </Button>
135     <Button x:Name="btnMoveDown" Command="{Binding Path = DataContext.ActivityMoveDownCmd,
136         RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}">
137         <Image Source="C:\Users\ * * * * \Documents\Visual Studio 2017\Projects\C3P0\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
```