Promasst MES

Bestandsnaam : MES-Tool-SAD

Versie : 0.5.0

Datum : 14-6-2018

Auteur : Koen Wartenberg

Software Architectuur Document

configuratie tool

# Document historie

## Versie beheer

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Versie** | **Beschrijving** | **Auteur** |
| 13-2-18 | 0.1.0 | Initiële versie | Koen Wartenberg |
| 26-2-2018 | 0.2.0 | Aanpassing EER-model | Koen Wartenberg |
| 29-3-2018 | 0.2.1 | Toevoeging parameter Gui design | Koen Wartenberg |
| 5-4-2018 | 0.3.0 | Database updates | Koen Wartenberg |
| 17-5-2018 | 0.4.0 | Documentatie is gelijk getrokken met het POC | Koen Wartenberg |
| 6-6-2018 | 0.5.0 | Documentatie update met huidige POC | Koen Wartenberg |
|  |  |  |  |

## Distributie

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Naam** | **0.1** |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## According document

|  |
| --- |
| Names  KSE Process Technology  DIVISIE 2: MARKETING & VERKOOP  Dep. 5 Product management  Erik Tenbult  Product owner  ………………… |

|  |
| --- |
| Namens  KSE Process Technology  DIVISION 4A: DEVELOPMENT  Dep. 11a Development verticals  Peter Noten  Contract manager  …………………. |

Inhoudsopgave

[1. Document historie 2](#_Toc514335176)

[1.1. Versie beheer 2](#_Toc514335177)

[1.2. Distributie 2](#_Toc514335178)

[1.3. According document 2](#_Toc514335179)

[2. Introductie 4](#_Toc514335180)

[2.1. Context 4](#_Toc514335181)

[2.2. Applicatie 4](#_Toc514335182)

[2.3. Doel van dit document 4](#_Toc514335183)

[2.4. Opbouw 4](#_Toc514335184)

[3. Logical view 5](#_Toc514335185)

[3.1. Architectuur 5](#_Toc514335186)

[3.2. Domein model 7](#_Toc514335187)

[3.3. Omschrijving tabellen 8](#_Toc514335188)

[3.4. UI-design 10](#_Toc514335189)

[4. Implementation view 11](#_Toc514335190)

[4.1. Implementatie model 11](#_Toc514335191)

[4.1.1. Diagram 11](#_Toc514335192)

[4.1.2. Beschrijving 11](#_Toc514335193)

[4.2. Database 12](#_Toc514335194)

[4.2.1. Database ontwerp 12](#_Toc514335195)

[4.2.2. Database benamingen 13](#_Toc514335196)

[4.3. Entity framework 14](#_Toc514335197)

[4.3.1. Klassen indeling 14](#_Toc514335198)

[4.3.2. Context 14](#_Toc514335199)

[4.3.3. Database persistentie 14](#_Toc514335200)

[4.4. Mappen structuur 15](#_Toc514335201)

[4.4.1. UI …………………………………………………………………………………………………………………………………………….15](#_Toc514335202)

[4.4.2. Business 16](#_Toc514335203)

[4.4.3. DAL 17](#_Toc514335204)

[5. Gebruikte Patronen 18](#_Toc514335205)

[5.1. MVVM (View-model-ViewModel) 19](#_Toc514335206)

[5.1.1. View model en interface 20](#_Toc514335207)

[5.1.2. Binding 21](#_Toc514335208)

[5.2. Singleton 22](#_Toc514335209)

[5.3. Repository (& entity framework) 22](#_Toc514335210)

[6. Exceptie handelingen 23](#_Toc514335211)

[6.1. Database Excepties 23](#_Toc514335212)

[6.2. Not Implemented Excepties 23](#_Toc514335213)

[6.3. Overige excepties 23](#_Toc514335214)

[7. Overige specificaties 24](#_Toc514335215)

[7.1. Code styling 24](#_Toc514335216)

[7.1.1. PascalCasing 24](#_Toc514335217)

[7.1.2. Underscore 24](#_Toc514335218)

[7.1.3. Afkortingen 24](#_Toc514335219)

[7.1.4. Keywords 24](#_Toc514335220)

[8. Bijlagen 25](#_Toc514335221)

[9. Bronnen 26](#_Toc514335222)

# Introductie

*In dit architectuur document worden aanwijzingen gegeven in italic (schuinschrift).*

*Rode tekst in dit document wijst erop dat iets nog niet af is (dit zal in de laaste versie eruit worden gehaald).*

## Context

De nadruk van dit document zal vooral gelegd worden op verbeteringen van het oude document. Hieronder vallen componenten die er eerste niet instonden en verbeteringen op het gebied van ontwerpen.

## Applicatie

PROMASST MESS configuratie tool is een applicatie waarmee batch prosessen geconfigureerd kunnen worden zodat het wegen, verpulveren, mengen en persen van voer automatisch makkelijk verloopt.

In de bestaande configuratie tool zitten nog verschillende bugs en design fouten. Ook is de code kwaliteit slecht en is er nauwelijks behulpzaam commentaar bijgeschreven.

## Doel van dit document

Het Software Architectuur Document (SAD) bevat een uitgebreide architecturale kijk op het systeem PROMASST MES Configuratietool dat verder ontwikkeld is door Koen Wartenberg voor KSE Process Technology. Het beschrijft een aantal verschillende architecturale views van het systeem om zo verschillende aspecten van het systeem te belichten.

## Opbouw

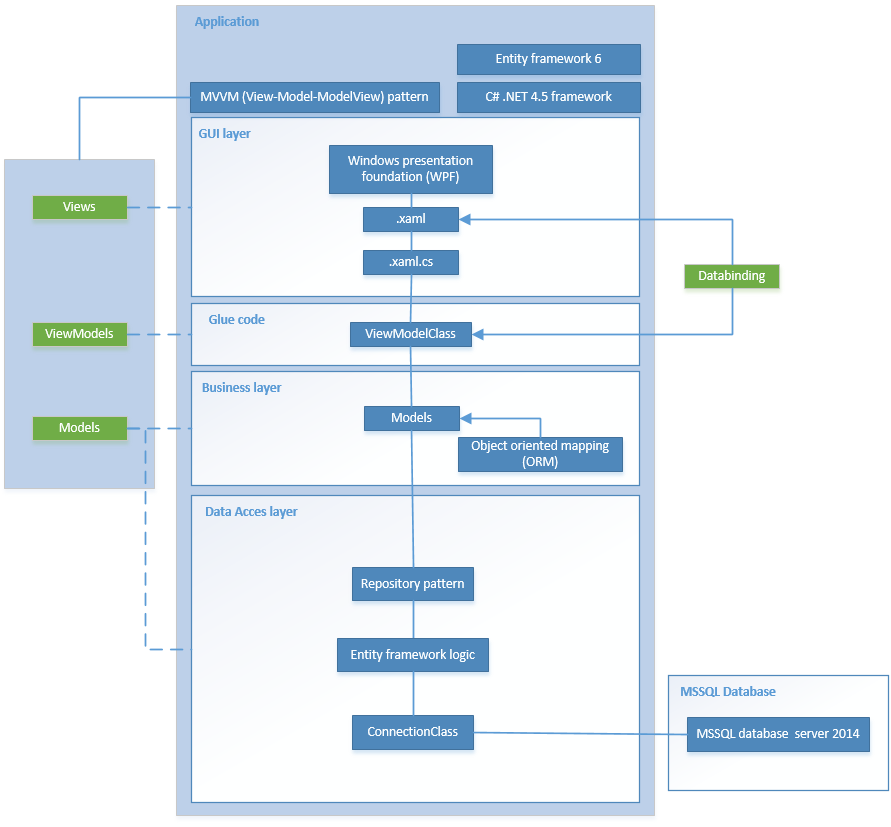
Dit document is opgebouwd uit Drie belangrijke hoofdstukken

* **Logical View**
  + Hierin zal er een algemene structuur van de applicatie geschetst worden
* **Implementation View**
  + Er zal hier dieper op de implementatie van de applicatie worden ingegaan. Er zal hier vooral gekeken worden naar opvallende implementatie keuzes of aanpassingen aan de database.
* **Patronen**
  + Hierin zullen de gebruikte patronen genoemd worden. Ook zal er een korte uitleg volgen over hoe ze zijn toegepast.

Als laatste zullen de exceptie handelingen nog een keer aan bot komen en de code styling.

# Logical view

## Architectuur



Figuur 1: architectuur diagram

Voor de applicatie is de drie lagen structuur toegepast. Er is duidelijk onderscheid gemaakt tussen de GUI, business laag en Data acces laag.

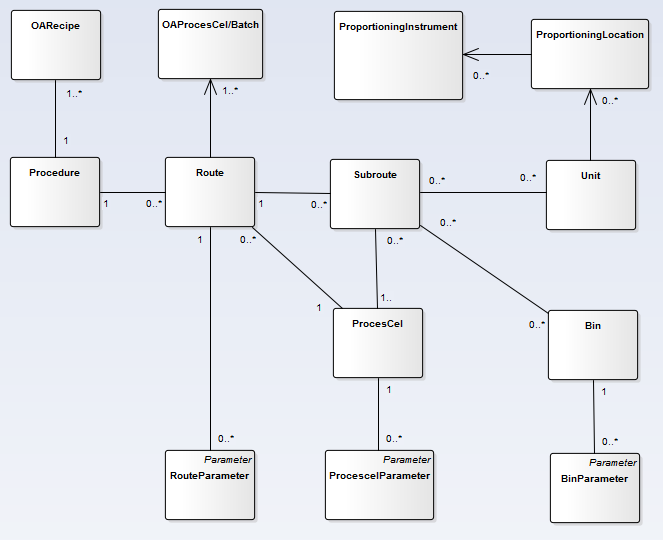
Er wordt in de applicatie gebruik gemaakt van het MVVM (Model-View-Viewmodel) patroon. Dit houdt in dat alles wat de gebruiker kan doen op de interface eerst terug gekoppeld wordt na de bijbehorende viewmodel die het weer doorgeeft aan de juiste klasse. Dit is de lijm code tussen de models.

Er wordt gebruik gemaakt van entity framework 6 in de applicatie. Bijna alle properties uit iedere class hebben dezelfde naam als een kolom uit de database. Of ze hebben een referentie daarna toe.

In de “Data Acces layer” word er gebruik gemaakt van een repository pattern in combinatie met entity framework. Door het repository patroon worden de verschillende database uitvoeringen overzichtelijk weergegeven en zijn fouten gemakkelijker terug te vinden.

Verder zal er geen gebruik worden gemaakt van Xml files. Deze zijn namelijk lastig te onderhouden en geven geen concreet overzicht van je opgeslagen data. Ook bied de database een beter alternatief.

## Domein model



Figuur 2: Domein model

Ga er bij deze afbeelding vanuit dat alles componenten die hier instaan in een fabriek gebeuren. Het wegen en persen en meer gebeuren natuurlijk allemaal door machines. Hieronder staat beschreven in volgorde wat elke component doet en hoe die in relatie staat tot de andere componenten.

Er staan in dit diagram alleen de belangrijkste componenten weergegeven waar de applicatie mee te maken heeft.

## Omschrijving tabellen

Procescell

Een onderdeel van de fabriek. Elk onderdeel is verantwoordelijk voor een deel van de verwerking van het product. De ene neemt bijvoorbeeld een grondstof in en de andere zorgt ervoor dat het vermalen wordt. Een procescel bestaat natuurlijk weer uit kleinere onderdelen hieronder beschreven.

Maar eerst enkele procescell soorten staan hieronder vermeld.

Dat zijn bijvoorbeeld:

* Innamelijn:

Neemt nieuwe grondstof materialen in voor gebruik binnen de fabriek.

* Menglijn:

Zorgt ervoor dat verschillende grondstoffen op de juiste manier samenkomen.

* Correctielijn:

Stel dat er van een stof te veel/weinig inzit in de huidige recept dan kan dat hier gecorrigeerd worden.

* Uitnamelijn:

Producten worden hier vervoert om opgeslagen te worden in een silo waarnaar ze kunnen worden

opgehaald.

* Perslijn:

Grondstoffen worden hier samengeperst.

* Transportlijn:

Aparte processCell die er voor zorgt dat de materialen van de ene processCell na de andere gaat.

Route

Een procescel bestaat uit een of meerdere routes. Het doel van een route is alleen om te bepalen welke subroutes in welke volgorde worden uitgevoerd.

Subroute

Een procescel heeft nul of meerdere subroutes. Deze routes kunnen daarna toegevoegd worden aan een route van de procescel. Een subroute kan meerdere keren voorkomen in een andere route van dezelfde procescel (dus niet in dezelfde route).

Unit

Een onderdeel van een processCell. Denk hierbij aan een weger en tijdelijke opslag cabine of een lopende band. Units worden weer toegevoegd aan een subroute. Een subroute kan meerdere units bevatten.

Procedure

Is een taak die binnen de fabriek door het systeem uitgevoerd moet worden.

OARecipe

Geeft aan hoe een taak uitgevoerd moet worden. Hier staat in hoeveel van elke grondstof gebruikt moet worden voor een bepaalt eindproduct en op welke manier de procedure uitgevoerd moet worden.

Bin

Een opslag plaats waar tijdelijk een product in bewaart wordt. Dit wordt ook wel een silo genoemd.

OAprocessCell/Batch

Stel je hebt een bepaald recept met daarbij drie keer de benodigde hoeveelheid om 1 eindproduct ervan te kunnen maken. Dan kun je dus drie keer het gewenste eindproduct maken. Een batch is een geautomatiseerd process die er voor zorgt dat een eindproduct doormiddel van een recept tot stand kan komen. In dit voorbeeld zullen er dus drie batches uitgevoerd worden.

Een route heeft minimaal een batch maar er kunnen er meerdere ingesteld worden.

ProportioningInstrument

Een instrument binnen een Unit die ervoor zorgt dat de juiste afwegingen van een product worden gedaan. Dit wordt dus alleen gebruikt bij bepaalde Units bijvoorbeeld Weegschalen.

ProportioningLocation

Een locatie van een of meerdere proportioning instrumenten.

Procescellparameter

Een procescel kan verschillende parameters bevatten een paar hiervan zullen benodigd zijn voordat er een aangemaakt. Deze parameters geven aan of er bepaalde zaken gecontroleerd moeten worden in de procescel of wanneer de procescel geen materiaal meer mag bevatten

Routeparameter

Een route kan verschillende parameters bevatten een paar hiervan zullen benodigd zijn voordat er een aangemaakt. Een route erft de meeste benodigde parameters over van de procescel waar die bij hoort.

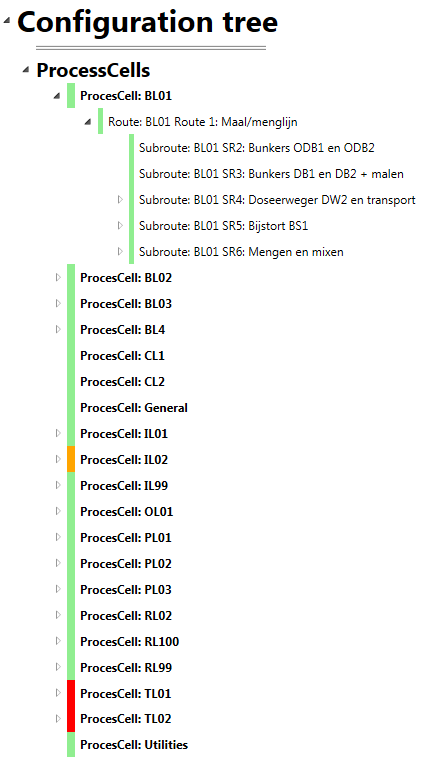
Binparameter

Een bin kan verschillende parameters bevatten een paar hiervan zullen benodigd zijn voordat er een aangemaakt. Deze parameter vermeld bijvoorbeeld hoeveel grondstoffen die mag of kan bevatten.

## UI-design

Het UI design zal er grotendeels hetzelfde eruit komen te zien qua opbouw en algemeen design. Er zal vooral gekeken worden in welke aspecten er verbeterd kunnen worden en welke aspecten weggehaald zullen moeten worden.

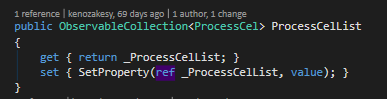
### Hoofdscherm



Figuur 3: configuratie boom structuur

De UI design voor de boom structuur is niet veel veranderd. In de code zit het MVVM patroon. Dit betekent dat als er in een gekoppeld element in de code veranderd dat dit dan te zien is in de user interface.

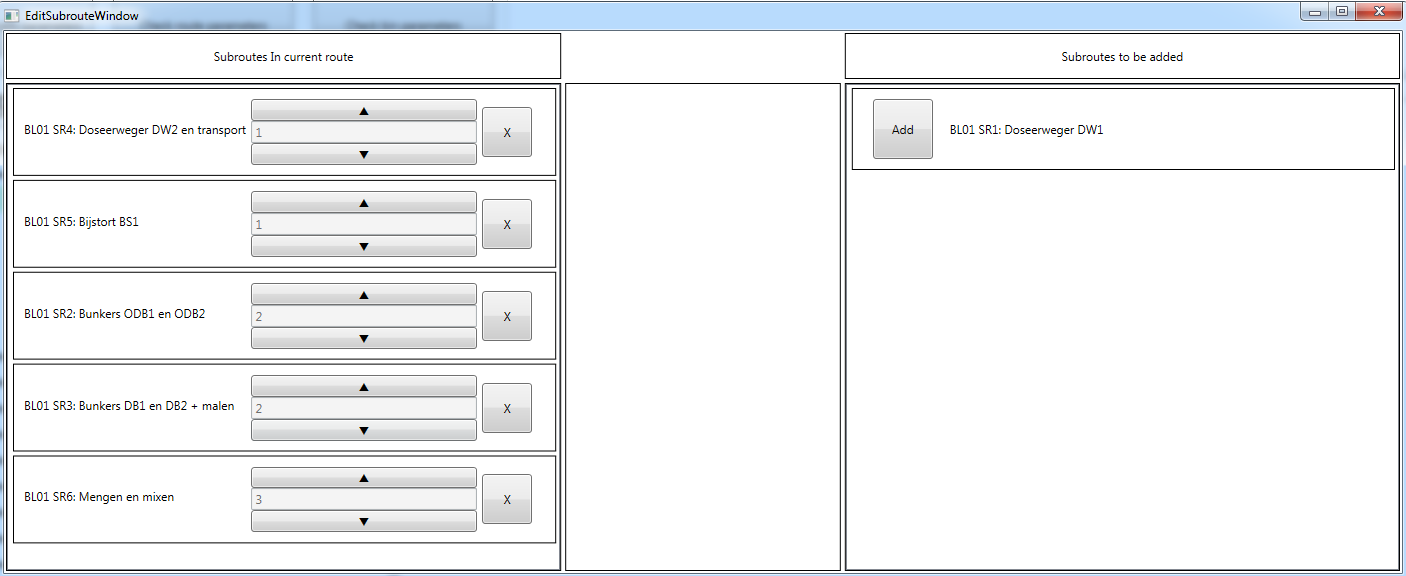
In dit geval zit de lijst:



Gekoppeld aan de treeview als de stam van de boom.

En in elke procescell zit weer een route lijst en in elke route zit weer een subroutelijst. Dit verhaal kan oneindig doorgaan als je zou willen.

### Set subroute scherm



5

4

3

2

1

Als je bij Routes klikt op “Set subroutes” dan komt het volgende scherm naar voren.

1. Dit is het vak waarin alle subroutes staan die de huidige route bevat.
2. Dit is waar alle subroutes staan die de route nog niet bevat.
3. De “Add” knop zorgt ervoor dat een subroute in de route wordt geplaatst
4. De “X” knop zorgt ervor dat de subroute uit de route wordt verwijderd (Hij blijft aanwezig in de procescell en andere routes)
5. Hier kun je de sequence nummer van de subroute in de route aan passen door op omhoog of omlaag te klikken.

IN dit proof of concept is er geen gebruikt gemaakt van een drag & drop scherm. Dit is gedaan omdat dit absoluut zeer onhandig is in combinatie met WPF MVVM. Daarom zijn er voor alternatieven gezocht. Deze user interface design is fuctioneel bruikbaar, maar voor gebruiksvriendelijkheid zit er nog veel verbetering.

# Implementation view

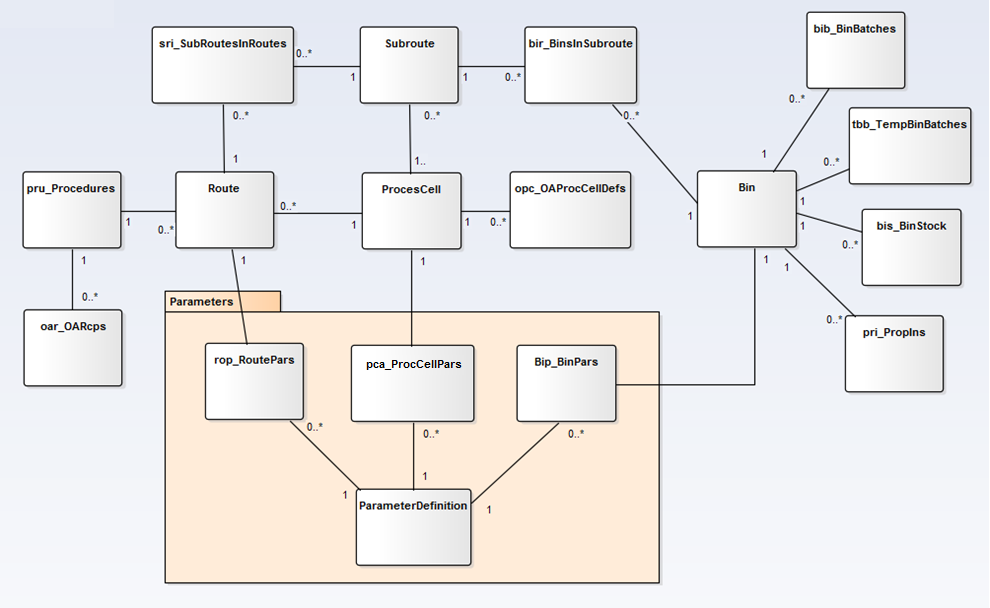
Niet alle klassen in de huidige applicatie worden gebruikt door het systeem. Sommige klassen staan alleen in de applicatie om die later te kunnen toepassen. Controleer dus zelf welke klassen je denkt nodig te hebben voordat je aanpassingen gaat verrichten.

Verder zullen alle klassen niet van al hun methodes voorzien zijn. Vooral de belangrijkste zullen aangegeven worden. De meeste methodes zijn nog een soort van testgevallen namelijk.

## 

## Implementatie model

### Business laag diagram (basis)

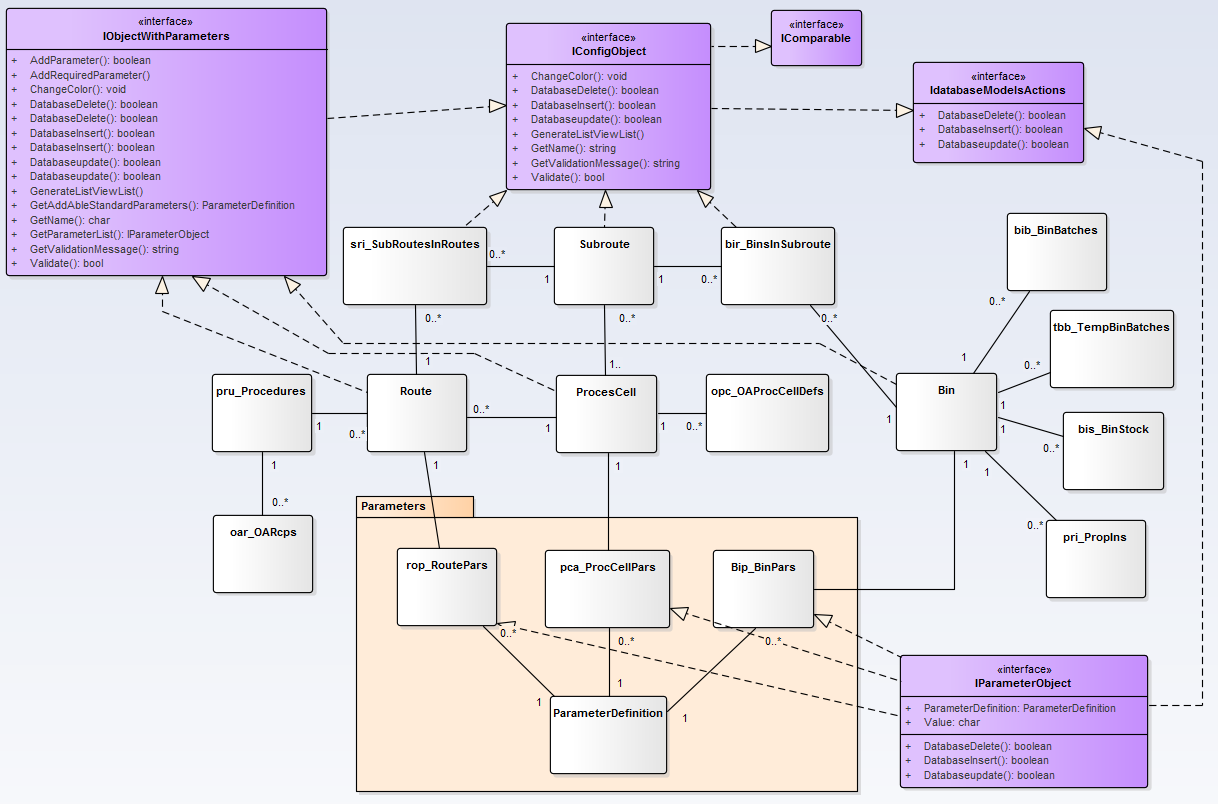


Figuur 4: Implementatie model

Dit is het algemene model van de applicatie. Dit model is opgebouwd aan de hand van entity framework. Het model is zo opgesteld dat de data gemakkelijk in een keer in de applicatie geladen hoef te worden zodat de applicatie tijdens runtime weinig meer zelf hoeft op te halen.

Alle klassen met “(drie letter)\_ + tabel naam” zijn klassen die bijna altijd direct uit de database zijn overgenomen voor entity framework. Er kunnen in de klassen zelf kleine aanpassingen zijn gedaan dus let hier wel op.

### Business laag diagram (met interfaces)



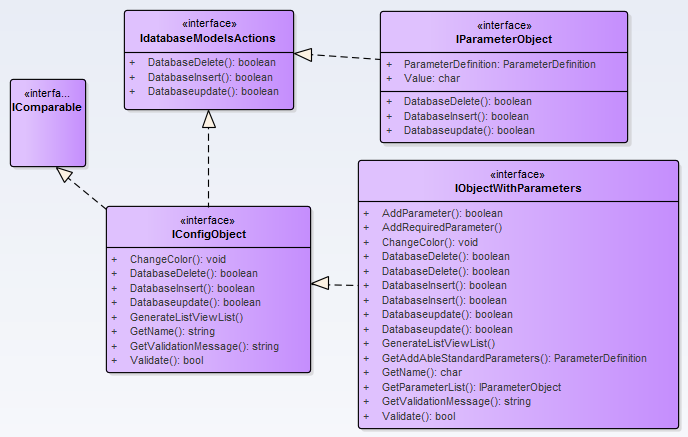
## Enums

In de onderste afbeelding zijn alle gebruikte enums van de applicatie te zien



Figuur ? : interfaces

## Interfaces



Figuur ? : interfaces

IDatabaseModelsActions

Hier staan alle database acties die een bepaald object moet kunnen om naar de database gezonden of opgehaald te worden.

IConfigObject

Deze interface wordt overgeërfd door bijna alle klassen in de applicatie. Het is een Interface die alle objecten in de objecten boom verzameld om ze uit te kunnen lezen

IObjectWithParameters

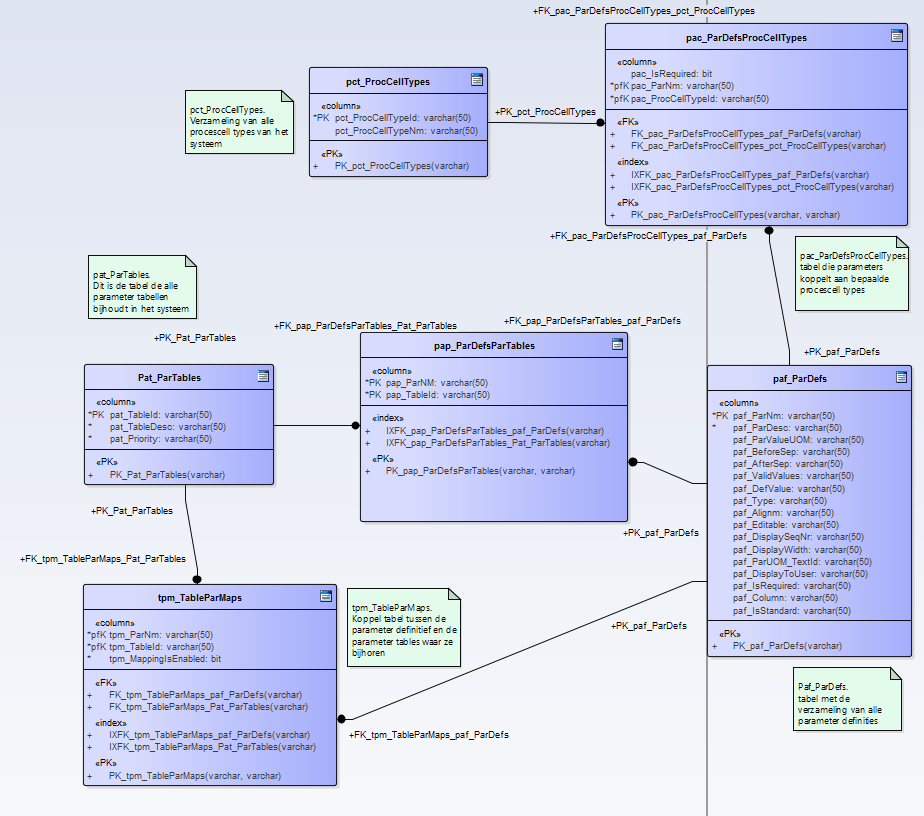
Deze interface wordt meegegeven aan objecten die parameters bevatten (bins, procescellen en routes).

IParameterObject

Deze interface wordt meegegeven aan alle parameters met uitzondering van de parameterdefinitie.

## Database

### Database ontwerp



Figuur 5: Database ontwerp (parameter configuratie deel)

In de database is er voor de het ophalen van de juiste parameters voor route en procescellen de PAC tabel aangemaakt. In deze tabel worden de parameter definities uit de PAF tabel gekoppeld aan een procescelltype uit de PCT tabel.

Daarnaast zal er gekeken worden of de parameter voor een bepaalt object in de applicatie verplicht is, optioneel is of juist helemaal niet geselecteerd kan worden als een potentiele parameter.

Andere type parameters kunnen nog niet verplicht worden gesteld. Er mag vanuit worden gegaan dat alleen de procescell en route objecten parameters toegekent kunnen krijgen.

### Database benamingen

*Tabel definities met een \* achter hun naam zijn koppeltabellen.*

Paf\_ParDefs

tabel met de verzameling van alle parameter definities

pct\_ProcCellType

Verzameling van alle procescell types van het systeem

pac\_ParDefsProcCellTypes \*

tabel die parameters koppelt aan bepaalde procescell types

pat\_ParTables

Dit is de tabel de alle parameter tabellen bijhoudt in het systeem

tpm\_TableParMaps \*

Koppel tabel tussen de parameter definitief en de parameter tables waar ze bij horen

pap\_ParDefsParTables \*

koppeltabel tussen de parameter tabel definitief en de parameter definities.

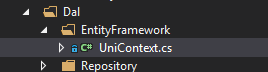
## Entity framework

### Klassen indeling

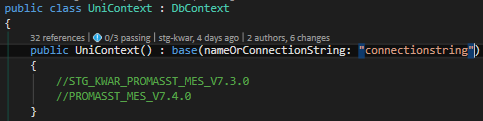
De klassen indeling is niet precies hetzelfde als de database.

(zie 4.1.1)

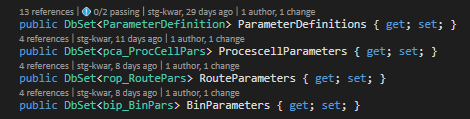
### Context



In de map entity framework staat de klasse “UniContext”. Dit is de context van entity framework en de daadwerkelijke klassen die in contact staat met de database.



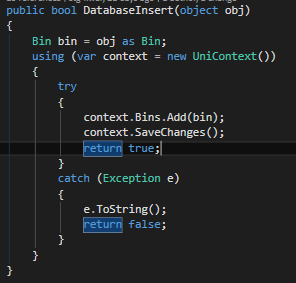
In de constructor schrijf je je connectionstring. Deze moet nog omgebouwd worden om contact te maken met PSTsystem.



Wanneer je een klassen hebt gelinkt (of als die dezelfde naam heeft als een database tabel) dan kan er een DBSet<> van gemaakt worden. Hierdoor kun je als programmeur de context van die klasse aanroepen en doormiddel van die klasse de database bewerken.

### Database persistentie

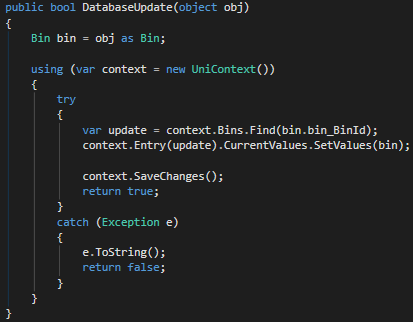
Insert



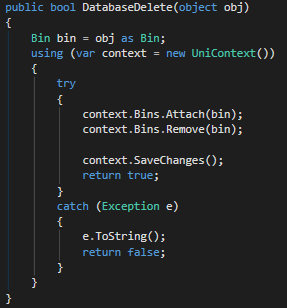
Je geeft een object mee aan de database repository. In dit geval is het een bin in de bin repository. Hier maak je een Bin van.

“Savechanges” zorgt ervoor dat de gegevens worden opgeslagen

Update



Delete

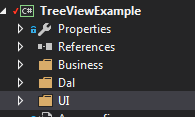


Bij een delete moet je eerst ervoor zorgen dat je het object dat je meegeeft aanhaakt aan het object in de database waar die bijhoort. Dit doe je door “attach” te gebruiken.

\*Mocht er iets fout gaan tijdens de database transactie dan wordt de gehele transactie afgebroken. Je kunt alles in de transactie door sturen naar de database of niks.

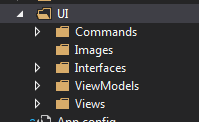
## Mappen structuur

De logische mappen structuur zal nu kort worden beschreven. Hierdoor zou het duidelijk moeten worden waar bepaalde klassen staan en hoe ze terug te vinden zijn.



Het project is opgedeeld in een Business laag, UI laag en een Dal laag

### UI



Commands

De Relaycommand gebruikt voor MVVM model

Images

Afbeeldingen gebruikt in de applicatie

Interfaces

Interfaces gekoppeld aan Views en viewmodels

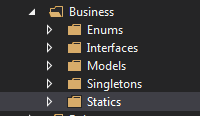
ViewModels

Viewmodels gebruikt in appliatie

Views

Alle Views (GUI schermen) staan hier

### Business



Enums

Enums gebruikt in applicatie

Interfaces

Interfaces gebruikt voor de Business laag

Models

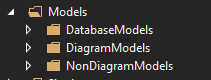
Classen gebruikt in de business laag.

Singletons

Classen met een singleton patroon

Statics

Alle static klassen in de applicatie of behaviours



DatabaseModels

Classen gebruikt met entity framework maar niet voor andere logica

DiagramModels

Classen gebruikt met veel implementatie logica. Vooral klassen die gebruikt zijn voor de treeview staan hierin.

NonDiagramModels

Dit zijn klassen die gebruikt worden om bijvoorbeeld een UI element er beter uit te later zien.

### DAL



Entityframework

Hierin staat de Databasecontext klasse

Repository

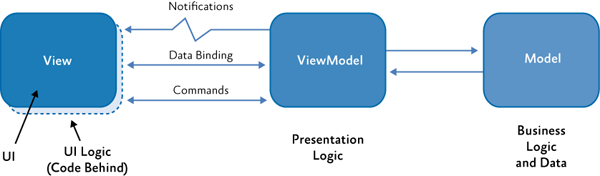
Hier is het repository patroon van de applicatie terug te vinden

# Gebruikte Patronen

|  |  |  |  |
| --- | --- | --- | --- |
| **Patroon** | **Gebruikt voor** | **Voordeel** | **Nadeel** |
| MVVM (Model-View-ViewModel) | Een duidelijke scheiding tussen GUI en Business laag. Wordt gebruikt in WPF applicaties | Duidelijke scheiding tussen busines en GUI laag. Is ook makkelijk te onderhouden | Redelijk Complex en niet waard om te gebruiken in kleine applicaties |
| Singleton | Het aanmaken van maar een instantie van een klas die overal aan te roepen is | Een stuk code kun je overal aanroepen en kun je gebruiken om gemeenschappelijke data in op te slaan | Code wordt snel onoverzichtelijk voor andere codeurs en moet daarom niet gebruikt worden als er andere mogelijkheden zijn. |
| Repository | Het communiceren met de data laag duidelijker te maken en de code daarvoor beter testbaar maken | Je code is beter testbaar en je kunt gemakkelijker van database wisselen. | Redelijk complex en niet waard om te gerbruiken in kleinere applicaties, tenzij je een versimpelde versie van maakt. |

In deze tabel is een snelle opsomming te zien van welke patronen gebruik zijn in de applicatie en de voor en nadelen en nadelen ervan. Hieronder is verder beschreven hoe deze uiteindelijk in de applicatie toegepast zijn.

## MVVM (View-model-ViewModel)



Figuur 6: MVVM patroon

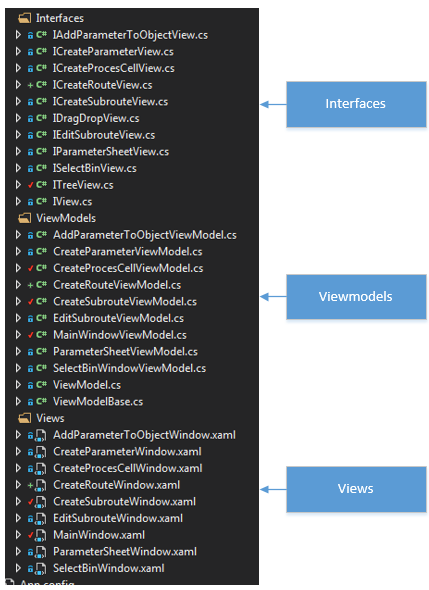
Door het MVVM patroon bestaat er een duidelijke scheiding tussen de GUI en Business laag in WPF projecten. Je kan de UI veranderen zonder dat je daardoor de business laag hoeft te veranderen of andersom (er zijn excepties houdt hier rekening mee).

### View model en interface

Wanneer je een nieuw scherm wil aanmaken dan heb je hier 3 componenten voor nodig.

Namelijk een:

* View
* Viewmodel
* Interface



Zorg er voor dat deze drie objecten dezelfde naamgeving hebben afgezien van of het een View, Viewmodel of interface is.

Wanneer deze drie elementen goed gekoppeld zijn dan kun je ze ook gaan gebruiken.

Zorg ervoor dat je geen UI logica zet in je ViewModels of models. Deze logica komt in de View te staan. Deze logica roep je aan door jouw aangemaakt interface object in de viewmodel.

logica zoals het laten zien van een pop-up scherm of naar een ander scherm gaan doe je in de View. Deze logica roep je weer aan vanuit je viewmodel

Kijk in de applicatie bij de map “UI” naar een voorbeeld hoe je deze drie elementen koppelt.

### Binding

Binding kan op twee manieren gedaan worden. Je hebt ten eerste command binding waarmee je methodes aan kunt roepen. En je hebt property binding. Hiermee verbind je een UI element direct aan een property in je viewmodel. Bijvoorbeeld textbox tekst aan een string.

#### Property binding



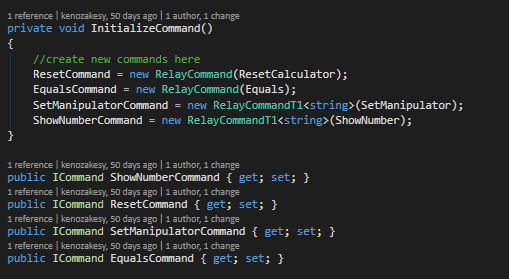
Met deze property binding kan bijvoorbeeld meteen deze tekst van bijvoorbeeld een textbox direct gekoppeld aan een property. Dit betekent dat als mijn property of mijn tekst in de textbox veranderd dan veranderd de andere ook.

#### Command binding

Command binding is iets ingewikkelder. Je moet eerst een command schrijven in bijvoorbeeld een “button” zoals hieronder.



Daarna zorg je ervoor dat je in je bijbehorende viewmodel het volgende property hebt staan.



De relaycommand staan in de map commands. Deze zorgen ervoor dat je een bepaalde methode in de viewmodel kunt aanroepen. Het type tussen “<type>” geeft een bepaalde parameter mee. Dit hoeft natuurlijk niet.

Als je wel een parameter mee wil geven zorg en dan voor dat je ook een command parameter mee geeft met je command



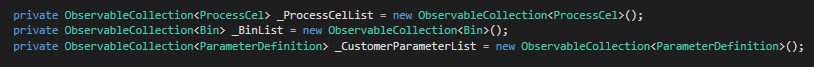
De “InitializeCommand” methode roep je aan vanuit je constructor.

## Singleton



De singleton “ListGodClass” wordt maar voor een doel gebruikt. Dat doel is om maar een instantie van elk database object te hebben en deze constant bij te kunnen houden. Hiermee zorg je ervoor dat er geen dubbele instantie in je applicatie terecht komen en ben je er van verzekerd dat elke uniek is.

Deze klasse houdt drie lijsten bij.



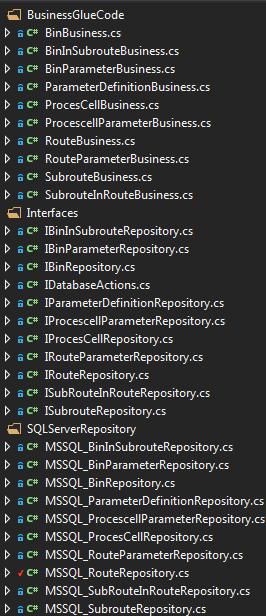
Niet alle objecten hoeven bijgehouden te worden aangezien de lijst met procescellen bijna alle data al bij houdt.

## Repository

Het repository patroon zit met entity framework in de applicatie verwerkt.



Elke klas heeft zijn eigen repository.



Een elke reposity bestaat uit 3 delen.

* De business lijm code
* Een interface
* De repository zelf die contact maakt naar de database.

Dus elke repository in deze applicatie bestaat uit:

* (naam) + Business.cs
* I + (naam) + Repository
* MSSQL\_ + (naam) + Repository

De beste manier voor het aanspreken van de DAL is dit.

Voorbeeld (procescell):

Bovenaan de klassen die je mee wil geven aan de database schrijf je het volgende:



Om nu de database aan te spreken doe je:



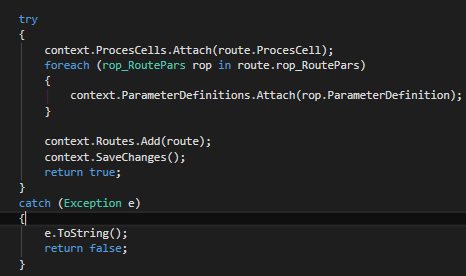
Doormiddel van deze manier toe te passen houw je een gemakkelijk overzicht van je database objecten. Maar ook weet je als programmeur waar en wanneer de database aangesproken wordt.

# Exceptie handelingen

Exception handelingen zullen op de volgende momenten in de applicatie plaatsvinden

## Database Excepties

Na elke bewerking op de database zal er een “try-catch-finally” gebruikt worden om mogelijke excepties af te vangen. De exceptie zal om het database statement zelf staan zodat mogelijk fouten meteen bij de bron ontdekt zullen worden.



Er wordt aangeraden om bij elke catch een breakpoint te zetten. Fouten in de database met entity framework zijn daardoor een stuk gemakkelijker op te vangen.

## Not Implemented Excepties

Elke keer wanneer er tijdens het programmeren niet meteen gebruik wordt gemaakt van een methode wordt deze ingevuld met “throw new NotImplementedException”.

## Overige excepties

Overige excepties zullen altijd met een “try-catch” moeten worden omringt. Daarnaast moet ook nog beschreven worden waarom er mogelijk een exceptie kan plaatsvinden.

# Overige specificaties

## Code styling

Voor de *code styling* zal er gebruikt worden gemaakt van de alternatieve C# benaming conventie.

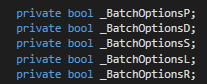
Er zal gebruik worden gemaakt van:

### PascalCasing

bij elke variabele naam zal het eerste woord een hoofdletter hebben. Ook elk daaropvolgende woord van een naam heeft een hoofdletter. (public void GetPersonId ())

### Underscore

Integendeel tot de normale C# conventie zal elke field naam beginnen met een underscore (\_) en daarna meteen een hoofdletter. Dit mag volgens de C# conventie nooit, maar het bedrijf houdt dit principe aan.



Figuur 7: voorbeeld fields benamingen

### Afkortingen

Liever leesbaarheid dan afkortbaarheid. “MoveX” is minder leesbaar dan “MoveHorizontal”.

Omdat er in de applicatie met entity framework gewerkt wordt zullen er veel afkortingen in de benamingen plaatsvinden. Ga dit niet veranderen. Het is namelijk een stuk consistenter wanneer de property namen overeen komen met kolom namen in de database.

### Keywords

Er zullen geen “keywords” worden gebruikt als benamingen of variaties daarop die conflicten opleveren met veel gebruikte programmeer talen.

Bron: Cwalina, K. (2008, oktober 22). *general-naming-conventions*. Opgehaald van docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventions

# Bronnen

Cwalina, K. (2008, oktober 22). *general-naming-conventions*. Opgehaald van docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventions