Eindverslag

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

Eindverslag

Ontwikkelen databasediagram generator voor Centric IT Solutions

Ted de Koning

REFERAAT

De Koning, T.F., Eindverslag van het afstudeerproject bij Centric IT Solutions, Juni 2010.

Dit verslag bevat een beschrijving van het proces dat de afstudeerder heeft doorlopen tijdens het uitvoeren van de afstudeeropdracht bij Centric IT Solutions.

Het afstudeerproject dient ter afronding van de opleiding Informatica aan de Haagse Hogeschool (HHS) en heeft als doel de kennis en vaardigheden van de student te bewijzen. Deze afstudeeropdracht is uitgevoerd door T.F. de Koning in opdracht van Centric IT Solutions. Het afstudeerproject is gestart op 8 februari 2010 en afgerond op 4 juni 2010.

Steekwoorden:

- .NET
- MVP
- DAL
- Oracle
- DBMS
- XML
- UML
- TTD
- AUP

VOORWOORD

Het afstudeerproject is uitgevoerd naar aanleiding van het afronden van mijn opleiding informatica aan de Haagse Hogeschool. Het project is uitgevoerd bij Centric IT Solutions, op de afdeling Belasting & Vastgoed.

Dit verslag is primair geschreven voor twee examinatoren van de Haagse Hogeschool en de gecommitteerde. Dit verslag is daarnaast bedoeld voor geïnteresseerden in mijn afstudeerproject. De lezer dient kennis te hebben met betrekking tot informatica.

Gedurende mijn afstudeerperiode was Patrick van Rijswijk de opdrachtgever en begeleider van het project. Ik wil hierbij Patrick van Rijswijk bedanken voor het mogelijk maken van het project en de ondersteuning ervan.

Ted de Koning
3 Juni 2010

Eindverslag

Ontwikkelen databasediagram generator voor Centric IT Solutions

Ted de Koning

INHOUDSOPGAVE

Eindverslag

Ontwikkelen databasediagram generator voor Centric IT Solutions

Ted de Koning

INLEIDING

Het doel van dit verslag is informatie en inzicht te bieden in het proces, de werkzaamheden die uitgevoerd zijn, de opgeleverde producten en de keuzes die gemaakt zijn binnen het afstudeerproject bij Centric.

Dit verslag is bedoeld voor de twee examinatoren van de HHS, een externe gecommitteerde, de opdrachtgever van het afstudeerproject en andere personen die in het afstudeerproject van mij geïnteresseerd zijn. Doordat het verslag technische begrippen met betrekking op software ontwikkeling bevat, is enige kennis van software ontwikkeling vereist om dit document goed te kunnen begrijpen.

Het eerste hoofdstuk beschrijft de bedrijfsorganisatie met daarin de positie van de student. De opdracht wordt beschreven in hoofdstuk twee. In hoofdstuk drie wordt de algemene aanpak van het project beschreven. Hoofdstuk vief tot en met zes bevatten de bevindingen in de verschillende fases binnen het afstudeerproject. Hoofdstuk zeven bevat informatie over de testactiviteiten.
In hoofdstuk acht wordt aandacht besteed aan reflectie.

Afsluitend bevat dit document enkele bijlagen.

1. BEDRIJFSORGANISATIE

Centric is een organisatie waar ongeveer 9000 mensen werkzaam zijn. Het bedrijf heeft vestigingen in meerdere landen binnen Europa, waaronder Duitsland, Noorwegen en Zweden.

Het bedrijf Centric is verdeeld in verschillende afdelingen, die onafhankelijk van elkaar werkzaam zijn. Elke afdeling heeft een eigen manager, deze manager is verantwoordelijk voor de producten en diensten die uitgevoerd worden binnen zijn/haar afdeling. Elke afdeling is opgedeeld in productgroepen, waarbij elke productgroep een eigen manager, technisch manager en
De opdrachtgever is de technisch-manager van de afdeling MVV binnen de afdeling Belasting & Vastgoed. De afdeling Belasting & vastgoed levert diensten voor overheidsinstanties, waardoor bijvoorbeeld processen om vergunningen te verlenen eenvoudiger en overzichtelijker worden.

De afstudeerder is werkzaam bij de afdeling MVV, op deze afdeling wordt software ontwikkeld waarmee gemeentes ondersteund worden in het proces van verlenen van vergunningen, waarbij onder andere rekening gehouden wordt met milieuwetten en omgevingsbeperkingen. De software is ontwikkeld in Delphi, maar wordt momenteel geconverteerd naar .NET. Op de afdeling wordt Oracle gebruikt als DBMS waar de applicaties mee werken.

2. OPDRACHT

2.1 AANLEIDING

Vanuit Centric is vraag naar een systeem waarmee de structuur van databases in diagramvorm weergegeven kan worden. De gegenereerde databasediagrammen kunnen worden gebruikt om maatwerk te vereenvoudigen en om systeembeheerders bij klanten te ondersteunen bij het aanpassen van een database.

2.2 UITGANGSSITUATIE

Centric IT Solutions levert diverse applicaties die Oracle databases gebruiken om data in op te slaan en te muteren. Sommige applicaties behoren tot een softwarepakket waarbij meerdere applicaties data uit verschillende databases gebruiken. De databases binnen het softwarepakket bevatten koppelingen naar elkaar om ervoor te zorgen dat dataduplicatie niet voorkomt. De data is gekoppeld door middel van Views die naar verschillende databases verwijzen.

2.3 PROBLEEMSTELLING

Op dit moment kunnen medewerkers van Centric alleen de structuur van databases bekijken door middel van de interface van het Oracle DBMS of een applicatie ontwikkeld door een derde partij, waarbij de broncode niet vrijgegeven is. Tevens komt regelmatig van de klant een verzoek om een databasediagram van de Oracle databases die bij de Centric applicaties horen te versturen, iets wat op dit moment niet mogelijk is. De applicaties waar Centric gebruik van maakt om de structuur van haar databases in te zien vragen SQL kennis van de gebruiker en bevatten een andere grafische interface dan de stijlgids van Centric voorschrijft.

2.4 DOELSTELLING

Tijdens de afstudeerperiode ontwikkelt de student een systeem waarmee databasediagrammen op basis van Oracle databases op het scherm getoond kunnen worden, om ervoor te zorgen dat ontwikkelaars bij Centric en systeembeheerders bij klanten inzicht in de structuur van Oracle databases krijgen.

2.5 MEERWAARDE EINDPRODUCTEN

Het doel van het ontwikkelde systeem is al in bestaande, commerciële systemen geïmplementeerd. Tijdens het project is door meerdere collega's gevraagd wat precies de meerwaarde van de eindproducten van mijn project is ten opzichte van bestaande softwarepakketten met hetzelfde doel.
Centric ontvangt regelmatig verzoeken van klanten om de databasestructuur van de geleverde applicaties in te zien. De opdrachtgever vindt dat het ontwikkelen van een applicatie die databasestructuren weer kan geven en de huisstijl van Centric heeft een betere uitstraling naar de klant toe geeft.
Het systeem is ontwikkeld in twee delen, waarbij een XML-bestand als databron gebruikt kan worden. Doordat data naar XML-bestanden geëxporteerd kan worden, is het mogelijk om data in te zien zonder een verbinding met een Oracle DBMS.

2.5.1 UITBREIDBAARHEID

De uitbreidbaarheid van het systeem is een belangrijk aspect voor Centric. De bronbestanden van het systeem zijn vrij beschikbaar voor Centric, waardoor toekomstig onderhoud mogelijk wordt. Door de gebruikte architectuur is het voor Centric mogelijk om delen van het realiseerde systeem in applicaties van Centric te integreren, waardoor gebruikers eenvoudig de database waar de desbetreffende applicatie invloed op heeft in te zien. Door de gebruikte architectuur is het tevens mogelijk om de twee applicaties binnen het systeem samen te voegen in één applicatie.
Het systeem kan ondersteuning voor de helpdesk van Centric bieden. Bij toekomstig onderhoud kan een koppeling naar het incidentensysteem van Centric gemaakt worden, waardoor de helpdesk database-incidenten aan een deel van een databasediagram kan koppelen en direct in het incidentensysteem kan plaatsen. Tevens kan toekomstig onderhoud notities bij database-elementen mogelijk maken, waardoor onderhoud op de database en ondersteuning op de database voor Centric eenvoudiger wordt.
Het systeem ondersteund momenteel enkel de Tabel, View en Function elementen van een database, Centric kan in de toekomst ondersteuning voor andere database-elementen, zoals procedures, toevoegen. Door de implementatie van de gedachte achter de Extractor applicatie (Zie hoofdstuk 4.6) is het mogelijk voor Centric om ondersteuning voor andere DBMS systemen aan het systeem toe te voegen.

Het ontwerp, de broncode en de tests zijn gedocumenteerd en beschikbaar voor Centric, waardoor onderhoud door software ontwikkelaars van Centric mogelijk is.

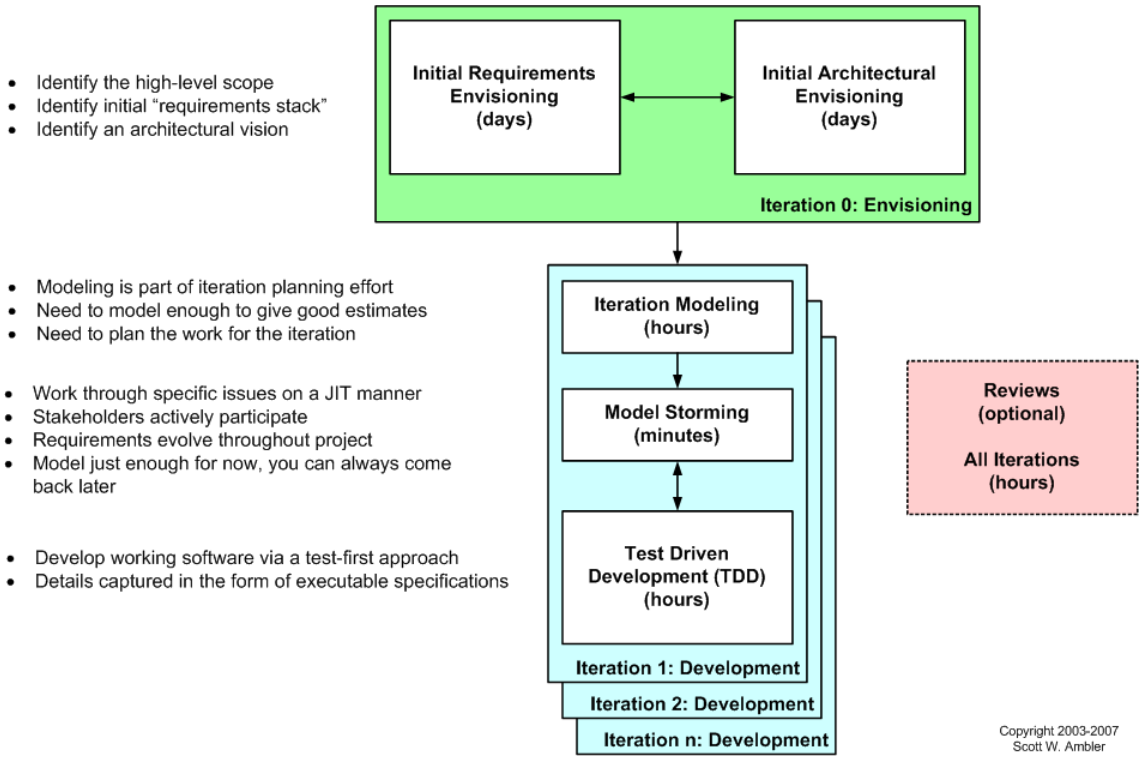
Door bovenstaande voordelen van het gerealiseerde systeem, krijgt het gerealiseerde systeem een betere overlevingskans binnen Centric.

3. AANPAK

3.1 ONTWIKKELMETHODE

Centric is op het moment van schrijven aan het experimenteren met verschillende Agile ontwikkelmethoden. De opdrachtgever stelde de eis dat voor dit project ook volgens een Agile ontwikkelmethode gewerkt zou worden. Na een korte evaluatie van verschillende Agile varianten, is ervoor gekozen om Agile Unified Process te kiezen.

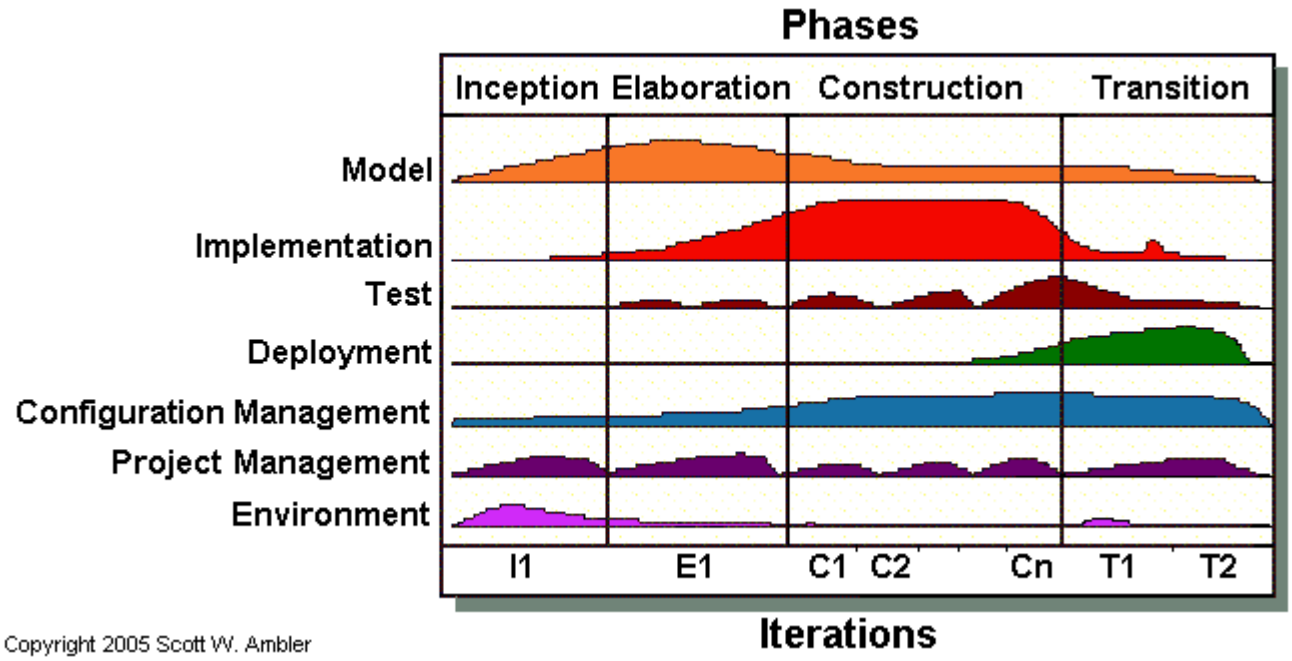
Agile Unified Process (AUP) is een Agile variant van het traditionele Unified Process (UP). AUP is een vereenvoudigde versie van UP, waarbij Agile technieken, zoals Test Driven Development, geïmplementeerd zijn. Agile ontwikkelmethoden richten zich vooral op de taken die echte waarde bevatten, zonder te veel op bijzaken te letten. Tevens wordt documentatie kort en bondig geschreven zodat het lezen ervan geen zware taak is.



Afbeelding 1, Ontwikkeling van een systeem volgens AUP

Afbeelding 1 toont het ontwikkelproces binnen AUP. Bij aanvang van een project wordt eerst een initieel ontwerp gemaakt, waarbij een globaal overzicht van de requirements en de architectuur gemaakt wordt. Zodra het initiële ontwerp is voltooid, wordt gestart met het iteratief implementeren van de requirements. Bij AUP wordt in iteraties gewerkt, waarbij eerst de requirements voor de te implementeren functionaliteit worden geanalyseerd en uitgewerkt worden. Nadat de requirements voor de iteratie vastgelegd zijn, wordt een ontwerp gemaakt, waarna het ontwerp door middel van Test Driven Development geïmplementeerd wordt.

3.1.1 SERIAL IN THE LARGE, ITERATIVE IN THE SMALL



Afbeelding 2, Verschillende fases en disciplines binnen AUP.

AUP bevat de verschillende fases die in UP gedefinieerd zijn. De verschillende fases zijn;

- **Inception fase**
- **Elaboration fase**
- **Construction fase**
- **Transition fase**

Net als bij UP, bevat elke fase een aantal disciplines waar per fase meer of minder aandacht aan besteed wordt. Volgens AUP worden de fases serieel doorlopen, terwijl disciplines per iteratie herhaalt worden om ervoor te zorgen dat voor elke iteratie een apart ontwerp is en getest wordt.

Tijdens het project zijn niet alle aspecten van AUP gebruikt. De transition fase is niet uitgevoerd omdat de opdrachtgever eerst het systeem laat verbeteren voordat het in gebruik genomen wordt.

3.1.1 JUST BARELY GOOD ENOUGH

Agile Unified Process (AUP) beschrijft dat artifacts zoals documenten en modellen op een bepaald punt voldoende zijn om zonder problemen te kunnen gebruiken. Met artifacts wordt projectdocumentatie en ontwerpmodellen bedoeld. De artifacts hoeven geen perfectie na te streven, maar zijn goed genoeg wanneer deze artifacts hun doel bereiken.

In tegenstelling tot andere methodes waarbij perfectie van elk artifact nagestreefd dient te worden, zorgt deze manier van werken voor tijdwinst bij het maken van de artifacts.
Just barely good enough is vergelijkbaar met het 80/20 principe, waarbij 80% van de waarde in 20% van de tijd toegevoegd wordt, maar de laatste 20% van de waarde in 80% van de tijd. In AUP is de laatste 20% niet noodzakelijk wanneer het artifact zijn doel bereikt heeft waardoor, in theorie, 80% van de tijd bespaard kan worden.

Op het moment dat een artifact zijn doel niet meer bereikt door een veranderde situatie moet deze aangepast worden tot het niveau dat het artifact zijn doel weer bereikt.

3.1.2 MODEL STORMING

Bij het gebruik van Agile Unified Process (AUP) wordt Model Storming gebruikt om de architectuur en functionaliteit te ontwerpen. Elke use case wordt vlak voordat het geïmplementeerd wordt, ontworpen. Bij traditionele ontwikkelmethodes wordt eerst een uitgebreid ontwerp gemaakt waarbij de architectuur en elk detail al vastgelegd wordt. Bij vrijwel elk ontwikkelproces veranderen eisen aan de applicatie of worden fouten in het ontwerp vastgesteld, om deze reden schrijft AUP de Model Storming techniek voor om ervoor te zorgen dat bij het initiële ontwerp van het systeem niet teveel detail vastgelegd wordt, er is immers meer kennis over de voor- en nadelen van het systeem wanneer het in ontwikkeling is. Deze manier van ontwerpen heeft tevens een ander voordeel, namelijk dat het software ontwikkelaars motiveert om mee te denken over het ontwerp, zodat niet enkel de software architecten hierover nadenken.

Tijdens het project is aanvankelijk een globaal ontwerp gemaakt waarbij weinig detail is vastgelegd. Per functionaliteit is gekeken hoe deze functionaliteit het beste binnen het ontwerp te implementeren is en welke bestaande functionaliteit gebruikt kan worden. Het initiële ontwerp is tijdens de ontwikkeling van het systeem meerdere malen gewijzigd totdat het uiteindelijke ontwerp dat in de komende hoofdstukken besproken wordt.

3.1.3 DOCUMENT LATE

AUP raadt aan om documentatie zo laat mogelijk te schrijven, op het moment waarop vrijwel alle benodigde informatie bekend is. Wanneer documentatie pas geschreven wordt wanneer vrijwel alle benodigde informatie bekend is, is meer kennis over het geïmplementeerde systeemdeel aanwezig en wordt in de documentatie minder gespeculeerd. Documentatie die zo laat mogelijk opgesteld is, bevat meer feiten. Door bovenstaande manier van werken, worden alle feiten op papier gezet nadat ermee gewerkt is, waardoor nog een keer naar alle informatie gekeken wordt en fouten op kunnen vallen voordat verder gewerkt wordt.

3.2 TECHNIEKEN

De te realiseren applicaties zijn in .NET ontwikkeld, waarbij gebruik is gemaakt van de ontwikkeltaal C#. Bij Centric worden alle nieuwe applicaties in C# ontwikkeld. Het ontwikkelen in .NET is uitgevoerd in de ontwikkeltool Visual Studio 2008. Het ontwikkelen van de applicaties is gedaan volgens Test Driven Development in combinatie met Model View Presenter, waardoor vrijwel elk stuk logica afzonderlijk te testen is.
Visual studio biedt ondersteuning voor unit-tests en kan alle beschikbare unit-tests met één druk op de knop uitvoeren.

Test Driven Development (TDD) is een techniek die door vrijwel elke Agile ontwikkelmethode gebruikt wordt om logica te testen. Unit-tests zijn ontwikkelaartests waarmee vastgesteld kan worden of de ontwikkelde functionaliteit werkt naar behoren en wanneer wijzigingen doorgevoerd worden, is het mogelijk om via unit-tests te controleren of de wijziging impact op andere elementen heeft gehad.

3.3 PLANNING

Onderstaande tabel toont de globale planning van het project.

Activiteit	Aanvang	Deadline
Opstellen use cases	week 1	week 2
Opstellen requirements	week 2	week 2
Opstellen domeinmodel	week 3	week 4
Ontwerpen sequence diagram	week 3	week 4
Ontwerpen klassediagram	week 3	week 4
Ontwerpen deployment diagram	week 3	week 4
Opstellen van testplan	week 4	week 5
Bouwen van applicatie (Extractor)	week 5	week 9
Maken van Unit-tests	week 5	week 14
Maken van PSD's	week 5	week 14
Bouwen van applicatie (Viewer)	week 10	week 14
Uitvoeren testplan	week 12	week 13
Opstellen eindverslag	week 13	week 17

Het bouwen van beide applicaties bevat korte iteraties, waarbij één use case geïmplementeerd wordt. Per iteratie wordt een detailontwerp ontworpen, unit-tests geschreven aan de hand van de use case en de functionaliteit geïmplementeerd. Tijdens de afstudeerperiode is de planning aangehouden.

4. INCEPTION FASE

4.1 AGILE UNIFIED PROCESS BIJ INCEPTION FASE

Tijdens de inception fase zijn de globale requirements opgesteld en zijn use cases ontworpen. De requirements zijn globaal gedefinieerd, wat volgens AUP dé manier is van ontwerpen. Nadat de requirements waren vastgesteld zijn de verschillende use cases gemaakt. De use cases bevatte in eerste instantie weinig detail en zijn tijdens implementatie uitgebreid met uitzonderingen en is de inhoud in kleine mate gewijzigd.

AUP definieert dat de gedachte achter de applicaties vastgelegd dienen te worden. Het nut van het definiëren van de gedachte achter de applicaties is dat functionaliteit die in de toekomst toegevoegd wordt, in de lijn van de applicatie dient te zijn, zodat de algehele functionaliteit van de applicatie logisch blijft.

4.2 OPSTELLEN PLAN VAN AANPAK

De eerste activiteit tijdens de afstudeerperiode is het ontwikkelen van een plan van aanpak.
Het plan van aanpak bevat details over onder andere de opdrachtomschrijving, de afbakening en de globale planning. Het plan van aanpak is opgesteld door het verwerken van data verkregen tijdens gesprekken met de opdrachtgever.

Het project is verdeeld in de verschillende fases van AUP. De planning is opgesteld door te kijken naar de verschillende producten die opgeleverd dienden te worden. Per product is er bekeken hoeveel tijd noodzakelijk is om het product af te ronden. Uiteraard kan een product naderhand nog gewijzigd worden op het moment dat een wijziging noodzakelijk is.

4.3 OPDELEN FUNCTIONALITEIT

Tijdens het opstellen van het afstudeerplan is bedacht om de functionaliteit van de softwarematige oplossing te scheiden. De oplossing is te verdelen in twee delen. Het eerste deel, de "Extractor", is het deel dat de metadata opvraagt uit de database. Deze applicatie zorgt ervoor dat een XML-bestand gegenereerd wordt aan de hand van de verkregen data.
Het tweede deel van het systeem is de "Viewer". De Viewer is een applicatie die ervoor zorgt dat een grafische weergave van metadata uit een database op het scherm getoond kan worden. De Viewer kan data uit, door de Extractor gegenereerde, XML-bestand inlezen.

Om verbinding te kunnen maken met een Oracle DBMS, is het noodzakelijk om Oracle client[1] of DotConnect[2] te installeren. Het is niet wenselijk om software van derden naar klanten te distribueren. Omdat er sprake is van gescheiden functionaliteit is het mogelijk om enkel de tweede applicatie naar klanten te distribueren.

[1] Software applicatie van Oracle om verbinding te kunnen maken met een Oracle DBMS

4.4 OPSTELLEN REQUIREMENTS

Het opstellen van de requirements is uitgevoerd aan de hand van informatie die gegeven is tijdens gesprekken met de opdrachtgever.
De opdrachtgever heeft een aantal eisen opgegeven aan het systeem, maar de afbakening werd aan mij overgelaten. Toen de afbakening van het project in het plan van aanpak gedefinieerd was, is de afbakening met de opdrachtgever besproken om ervoor te zorgen dat er geen miscommunicatie ontstond.

4.4.1 EENVOUD

Om onderhoudbaarheid en flexibiliteit te bevorderen is enkel de functionaliteit in de requirements opgenomen die noodzakelijk is om de use cases te implementeren. Volgens Agile Unified Process moet een applicatie zo eenvoudig en modulair mogelijk ontworpen worden en moeten bijzaken achterwege gelaten worden. In sommige gevallen zijn uitzonderingen echter onvermijdbaar, zoals het verplaatsen van elementen binnen het databasediagram.
Het verplaatsen van elementen binnen het databasediagram is geen primaire functionaliteit, maar omdat veel diagrammen niet volledig leesbaar gegenereerd kunnen worden, is het noodzakelijk in deze situatie.

4.5 MAKEN VAN USE CASES

De use cases zijn opgesteld aan de hand van de eisen die vastgelegd zijn in de requirements. Per applicatie is een use case diagram gemaakt met daarin de te realiseren functionaliteit. De use cases zijn beschreven met de mogelijke uitzonderingen en resultaat.

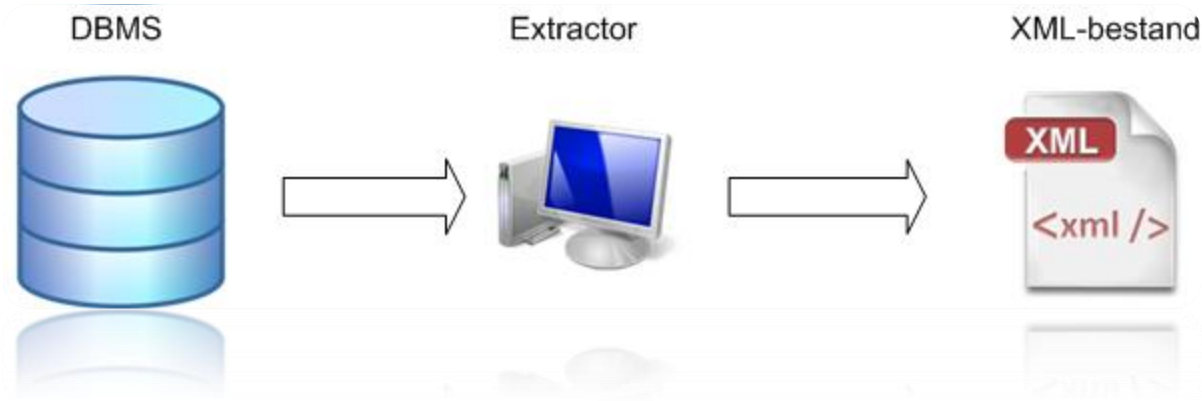
Bij het opstellen van de use cases is erop gelet dat enkel de noodzakelijke functionaliteit hierin opgenomen zou worden. In overleg met de opdrachtgever is ervoor gekozen om eerst te concentreren op de meest noodzakelijke functionaliteit. Extra functionaliteit wordt alleen geïmplementeerd wanneer daar voldoende tijd voor over is.

[2] Software applicatie van Devart om verbinding te kunnen maken met een Oracle DBMS.

4.6 GEDACHTE ACHTER APPLICATIES

Volgens AUP moet een systeem ontwikkeld worden met een achterliggende gedachte over de functionaliteit van het systeem. Binnen het gerealiseerde systeem zijn twee gedachtes bedacht bij de twee applicaties. De gedachtes worden gebruikt om te bepalen of bepaalde functionaliteit binnen de applicatie hoort en op de juiste manier te implementeren is.

4.6.1 EXTRACTOR

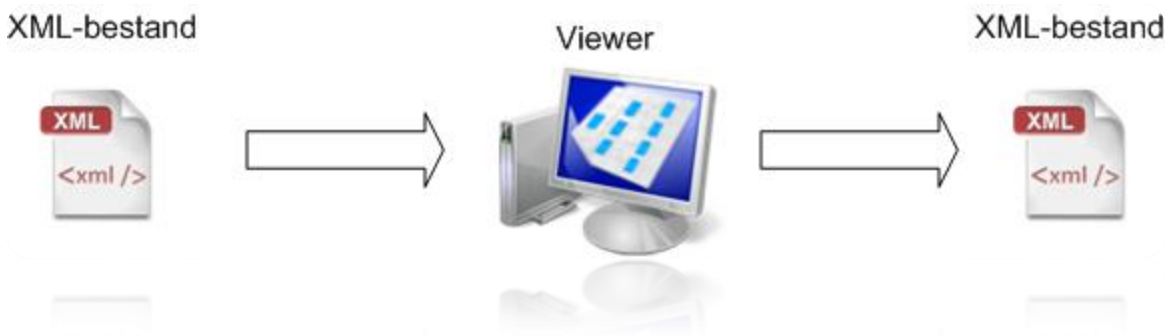


Afbeelding 3, Visuele weergave van gedachte achter de Extractor applicatie.

De gedachte achter de Extractor applicatie is het omzetten van data vanuit een databron naar een ander formaat. Elke Use case heeft een eigen Model View Presenter (MVP) instantie en heeft geen koppeling met andere MVP instanties.
Om toekomstig onderhoud eenvoudiger te maken staan alle importeer en exporteer mogelijkheden los van het Model, waardoor Loose Coupling ontstaat.
Wanneer de opdrachtgever de applicatie uit wilt breiden met ondersteuning voor SQL Server of een ander DBMS, is geen wijziging in de huidige source code nodig.

Met bovenstaande gedachte in het achterhoofd is gezocht naar design patterns waarbij deze vorm van ontwikkelen ondersteund wordt.

4.6.2 VIEWER



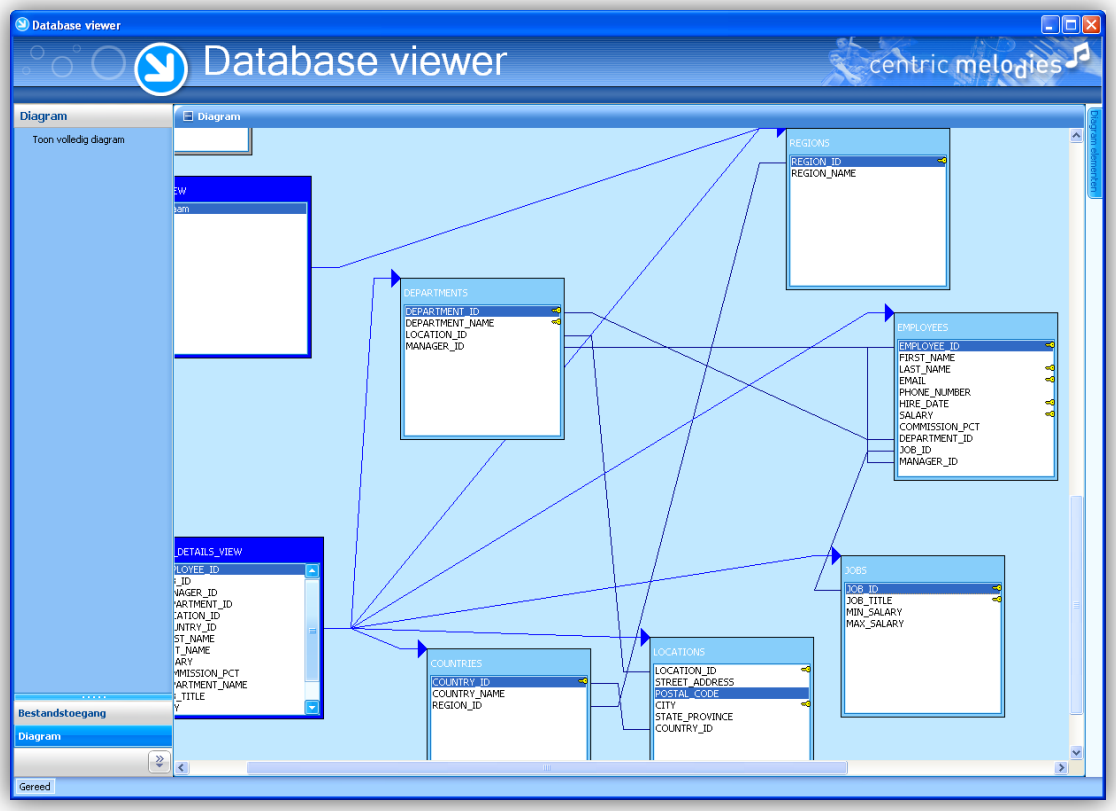
Afbeelding 4, Visuele weergave van gedachte achter de Viewer applicatie

De Viewer is ontworpen en ontwikkeld met de gedachte om, door de Extractor geëxporteerde, data te kunnen tonen op het scherm. De data wordt als databasediagram getoond aan de gebruiker. De Viewer maakt gebruik van dezelfde logica als de Extractor tijdens het importeren van- en het schrijven naar XML-bestanden.

4.7 ONTWERPEN GEBRUIKERSINTERFACE

De opdrachtgever wenst dat de gebruikersinterface van het systeem volgens de Centric stijlgids gebouwd wordt. De Centric stijlgids definieert een aantal eisen waaraan schermen binnen applicaties aan moeten voldoen. Zo moet elk systeem een navigatiebalk, die door Centric ontwikkeld is, bevatten waarmee tussen verschillende schermen geschakeld kan worden. Het systeem moet bovenaan het scherm een blauwe titelbalk hebben waarin de titel van de applicatie staat, deze titelbalk is tevens door Centric ontwikkeld.

Formulieren en standaard elementen moeten gebruik opgesteld worden door gebruik te maken van de Centric Controls, dit zijn elementen die de Centric huisstijl bevatten en veelgebruikte functionaliteit ondersteunen.
De Viewer heeft een scherm waarbij geen niet geheel gebruik van de Centric Controls gemaakt kan worden, dit scherm is het scherm waarbij diagrammen getoond worden. Het scherm waarbij diagrammen getoond worden implementeerd de Centric stijlgids door gebruik te maken van de kleuren die in de stijlgids gedefinieerd zijn. Het ontwerp voor het diagramscherm is opgesteld na gekeken te hebben naar applicaties zoals Microsoft Visio en Smartdraw. Nadat naar andere applicaties is gekeken is een voorlopig ontwerp gemaakt. Het eerste ontwerp is in de loop van de ontwikkeling van het systeem aangepast, om uiteindelijk op het onderstaande ontwerp uit te komen.



Afbeelding 5, Uiteindelijke gebruikersinterface van Viewer.

5. ELABORATION FASE

5.1 AGILE UNIFIED PROCESS BIJ ELABORATION FASE

In de elaboration fase is vooral aandacht besteed aan het ontwerpen van een globale architectuur voor het systeem. Het ontwerpen van een architectuur van het systeem is gedaan volgens de gedachte dat elke functionaliteit los staat van de ander. Eigenlijk is elke functionaliteit die geïmplementeerd is, een aparte iteratie die toegevoegd wordt aan het systeem. Tijdens de elaboration fase is vooral aandacht besteed aan de architectuur van het systeem en hoe de verschillende Use cases onafhankelijk van elkaar geïmplementeerd kunnen worden.

AUP definieert dat de ontwikkeling van een systeem op iteratieve wijze gedaan dient te worden. Bij de ontwikkeling van het systeem tijdens de afstudeerperiode is elke use case apart ontworpen en geïmplementeerd, waardoor elke use case een aparte iteratie is.

5.2 APPLICATIE-ARCHITECTUUR

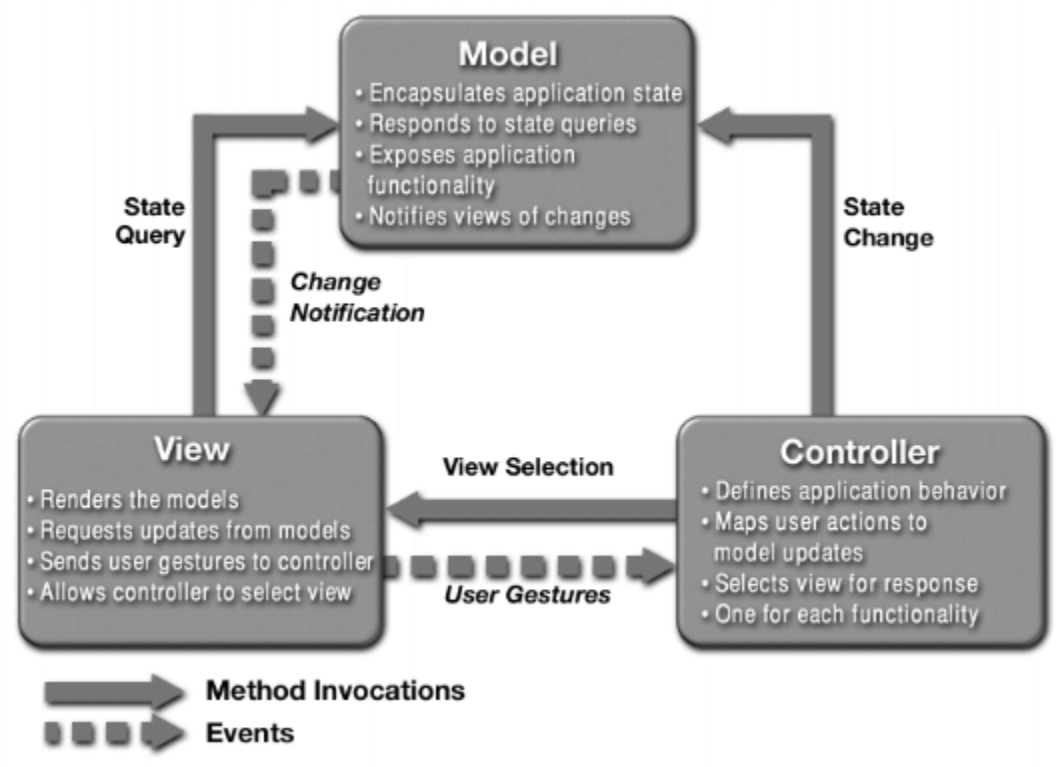
De opdrachtgever wenst dat de functionaliteit van beide applicaties uitbreidbaar en herbruikbaar zijn. Om te kunnen zorgen dat de functionaliteit herbruikbaar is, is het noodzakelijk om een juiste architectuur te implementeren.
Van te voren is bedacht dat de user interface gescheiden moet zijn van de onderliggende logica. Het scheiden van de user interface met de onderliggende logica zorgt ervoor dat de logica herbruikbaar is.
Op het internet is gezocht naar verschillende architecturen, waarbij een applicatie in verschillende lagen gedeeld wordt.

5.2.1 ARCHITECTUUR KANDIDATEN

MODEL VIEW CONTROLLER

Model View Controller is een software architectuur bestaande uit drie delen. Het eerste deel is het Model, dit deel definieert de data representatie in de applicatie. Het Model wordt gebruikt om data tijdelijk in op te slaan die bijvoorbeeld door de gebruiker is ingevuld of uit een data bron gehaald is.
Het tweede deel is de View. De View is representeert de gebruikers interface waarin informatie aan de gebruiker wordt aangeboden. De View heeft een directe koppeling met zowel de Controller en het Model.
Het derde deel is de Controller. De Controller zorgt voor verwerking van data vanuit de View en slaat deze op in het Model. De Controller bevat ook de applicatielogica.

Het Model heeft geen koppeling met de View, waardoor nieuwe Views eenvoudig toegevoegd kunnen worden.



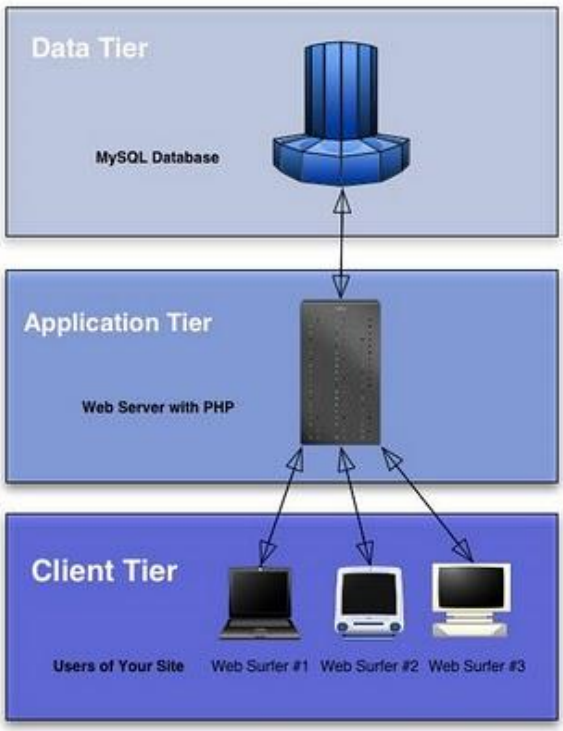
Afbeelding 6, Illustratie van de MVC architectuur[3]

[3] Bron: http://www.siliconwhisperer.com/index.php/tag/3-tier/

THREE-TIER

De Three-tier architectuur bevat, net als Model View Controller, drie lagen om de functionaliteit van de applicatie te scheiden. Three-tier bestaat uit een Presentation-tier, een Application-tier en een Data-tier. De Presentation-tier geeft de informatie aan de gebruiker weer.
De Application-tier zorgt voor de applicatielogica en de verwerking van data vanuit de Presentation-tier.
De Data-tier communiceert met een database of een andere databron.

De Three-tier architectuur heeft veel gemeen met de MVC architectuur. Het verschil tussen Three-tier en MVC zit in de communicatie tussen de gebruikersinterface en de datalaag. In Three-tier gaat alle communicatie van de Presentation-tier naar de Data-tier via de Application-tier, terwijl bij MVC de View direct kan communiceren met het Model.



Afbeelding 7, Voorbeeld van een Three-tier implementatie[4]

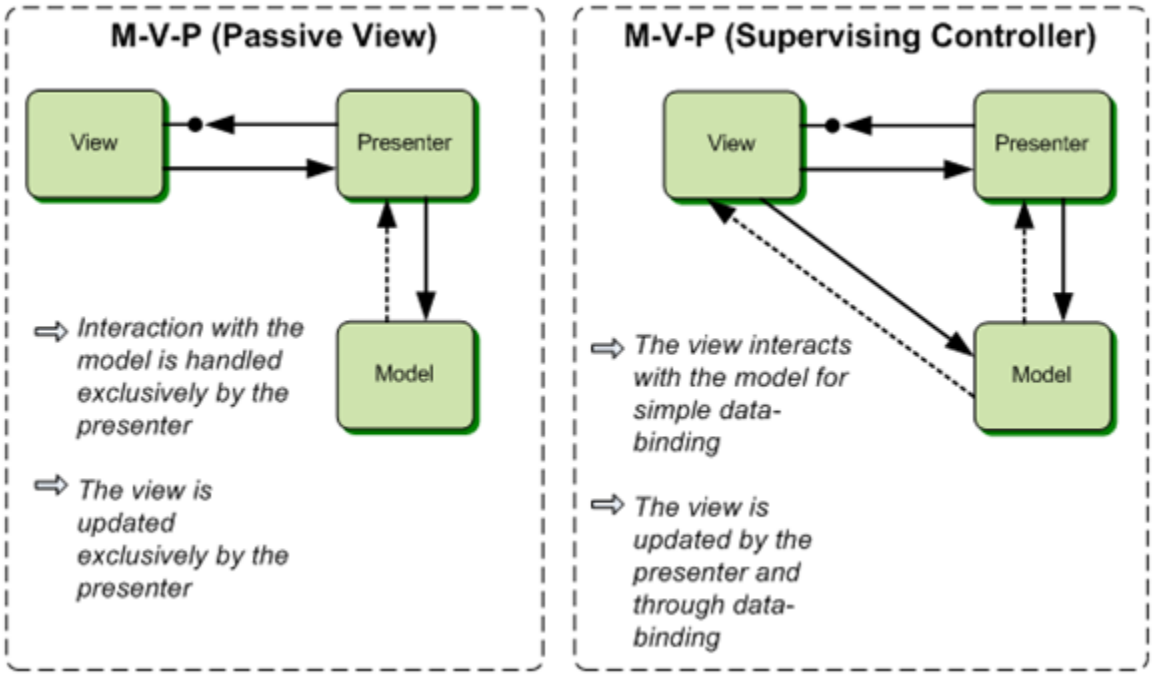
[4] Bron: http://siliconwhisperer.com/2007/12/three-tier-web-application-framework/

MODEL VIEW PRESENTER

Model View Presenter (MVP) is tevens een architectuur met drie lagen. MVP bestaat uit een Model, een View en een Presenter. Het Model bevat, net als bij MVC, de tijdelijk opgeslagen data die door de gebruiker ingevoerd is of opgehaald is uit een databron.
De View bevat de gebruikersinterface die de gebruiker op het scherm te zien krijgt.
De Presenter bevat de applicatielogica en zorgt voor de afhandeling van acties van de gebruiker.

MVP heeft twee varianten, Passive View en Supervising Controller.
Passive View definieert dat de View altijd communiceert met de Presenter. De Presenter communiceert op zijn beurt weer met het Model. Deze implementatie heeft veel weg van de Three-tier architectuur.
Bij Supervising Controller kan de View met zowel de Presenter als met het Model communiceren. De communicatie tussen de View en de Model bestaat uit eenvoudige data-binding. Ingewikkelde handelingen wordt via de Presenter afgehandeld. Deze implementatie is vrijwel identiek aan de MVC architectuur.

MVP verschilt met bovenstaande architecturen doordat het voor elke Use case een aparte MVP instantie bevat waardoor elke Use case onafhankelijk van een ander geïmplementeerd kan worden.



Afbeelding 8, Illustratie van de twee MVP varianten[5]

[5] Bron:http://msdn.microsoft.com/en-us/library/cc304760.aspx

5.2.2 GEKOZEN ARCHITECTUUR

Uiteindelijk is de keuze gemaakt om Model View Presenter (MVP) te gebruiken. De architectuur is door mij gekozen en is gecontroleerd en goedgekeurd door een software architect bij Centric. MVP zorgt ervoor dat de gebruikersinterface volledig los staat van de onderliggende logica en zorgt voor herbruikbare en onderhoudbare logica.

Er is gekozen om Passive View te gebruiken voor beide applicaties. De reden voor deze keuze is het eenvoudig maken van de applicaties. Data-binding is op dit moment niet noodzakelijk en wanneer er gekozen wordt voor Supervising Controller geeft dat een extra mogelijkheid om vanuit de View data op te halen waardoor er twee manieren zijn om data in de View te krijgen. Na overleg met een software architect van Centric is door mij de keuze gemaakt om Passive View te gebruiken.

Nadat bekend gemaakt werd dat MVP gebruikt wordt in beide applicaties kreeg ik te horen dat dat deze architectuur ook gebruikt wordt in applicaties van Centric. Doordat Centric gebruikt maakt van MVP in haar applicaties, past deze architectuur tevens het beste bij het gerealiseerde systeem.

5.2.3 BUSINESS LOGIC LAYER

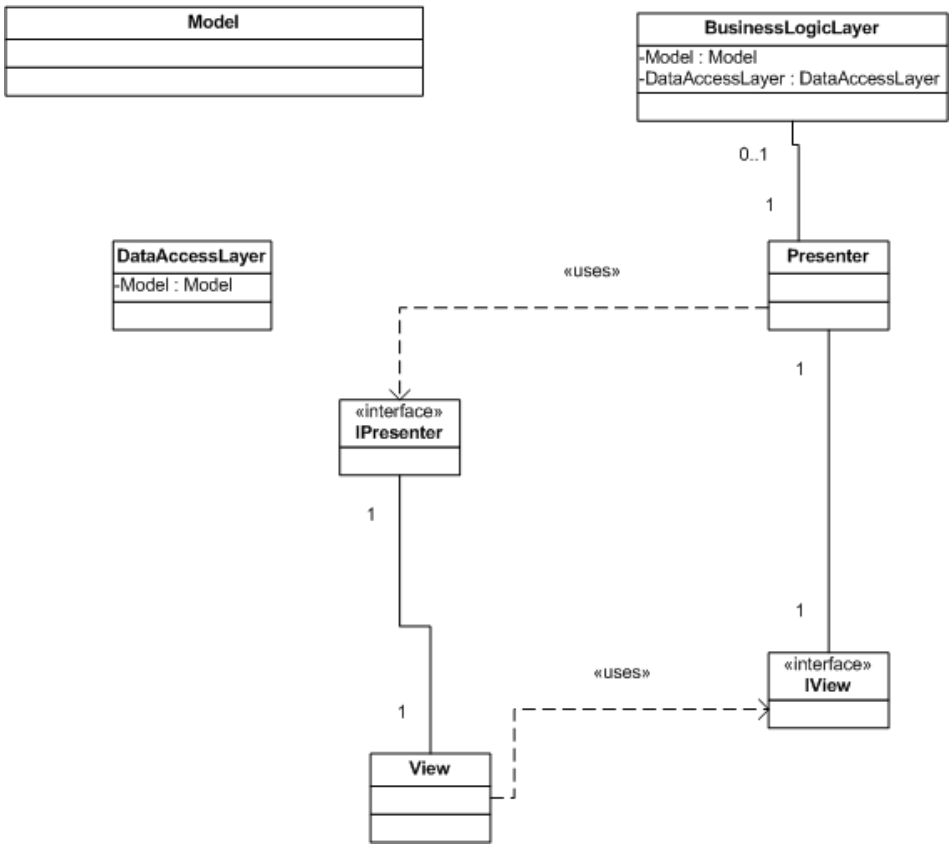
Naast MVP bevatten de applicaties per Use case een Business Logic Layer (BLL). Het BLL bevat de logica benodigd om de Use cases te implementeren. Doordat de BLL deze logica uit handen neemt van de Presenter, wordt de Presenter niet veel meer dan een verificatielaag tussen de logica en de gebruikersinterface. Door de deling van functionaliteit is het nog eenvoudiger om de applicatie te onderhouden. De Business Logic Layer is niet bedoeld als gedeelde functionaliteit en is slechts relevant op één use case.

5.2.4 DATA ACCESS LAYER

Naast de implementatie van MVP en het implementeren van een BLL is nog een laag aan de applicaties toegevoegd, de Data Access Layer (DAL). Het DAL bevat functionaliteit om data te kunnen importeren en exporteren. Het doel van het DAL is ervoor zorgen dat het Model geen relatie heeft tot welke importeer of exporteer functionaliteit dan ook en dat de logica herbruikbaar is in verschillende use cases en in beide applicaties. Wanneer Centric een nieuw DBMS gaat ondersteunen, moet het immers mogelijk zijn om het systeem met zowel Oracle als met het nieuwe DBMS te kunnen laten werken. Elke class binnen het DAL moet onafhankelijk zijn en alleen een referentie hebben naar het Model.
Het DAL is bedoeld als gedeelde functionaliteit, zodat beide applicaties van dezelfde logica gebruik kunnen maken. De connectie naar Oracle is niet geïmplementeerd in het DAL omdat de Viewer geen databaseconnectie nodig heeft en de gebruiker nu geen DotConnect dient te installeren om de Viewer te kunnen gebruiken. De DAL bevat wel de interface naar de databaseconnectie, zodat eventueel later een databaseconnectie toegevoegd kan worden.

5.2.5 CLASSDIAGRAM GEKOZEN ARCHITECTUUR

Onderstaand diagram toont de relatie tussen de lagen in de applicatie, inclusief de DataAccessLayer en de BusinessLogicLayer en is bedoeld om een globale indruk van de structuur in het systeem te tonen.
Elke implementatie van een use case is opgebouwd aan de hand van deze structuur, waardoor de use cases onafhankelijk van elkaar zijn, maar onderhoudbaar blijven doordat dezelfde structuur gebruikt wordt.
Een instantie vanuit de DataAccessLayer is een importeer- of exporteerclass, die gebruikt kan worden voor communicatie naar een Oracle DBMS of een XML-bestand.
Het BusinessLogicLayer bevat de logica benodigd voor de implementatie van één use case.
Een BusinessLogicLayer hoeft niet altijd een DataAccessLayer te hebben, enkel in de situatie van importeren en exporteren is dit noodzakelijk.



Afbeelding 9, Diagram van de globale architectuur van het systeem.

5.3 UNIT-TESTS

Agile Unified Process definieert dat ontwikkelen moet gebeuren volgens de Test Driven Development (TDD) methode. TDD is een methode waarbij per functionaliteit eerst Unit-tests ontwikkeld worden voordat de daadwerkelijke logica geïmplementeerd wordt. Het voordeel van TDD is dat bij onderhoud van de applicatie de tests een controlerende rol spelen. Wanneer een functionaliteit aangepast wordt en een test faalt geconstateerd kan worden dat de wijziging op een andere plaats binnen het systeem effect heeft gehad.
Elke Unit-test staat los van de omgeving waar de test uitgevoerd wordt, waardoor de tests op elke PC met windows uitgevoerd kunnen worden.

Door het gebruik van MVP is Unit-testen van alle logica mogelijk. Doordat elke Use case individueel is geïmplementeerd, is het mogelijk om de verschillende situaties goed te testen.

Visual Studio biedt ondersteuning bij het ontwikkelen van unit-tests door het mogelijk te maken om alle tests voor een project in één keer uit te voeren en bevat functionaliteit om waardes met elkaar te vergelijken.
Door gebruik te maken van het Code Coverage rapport van Visual Studio, is eenvoudig na te gaan of alle paden binnen een functionaliteit getest worden.

5.4 TESTPLAN

5.4.1 TESTMETHODE

De hanteerde testmethode tijdens het afstudeerproject is TMap. TMap staat voor Test Management approach en is ontwikkeld door Sogeti. TMap is een methode voor het beheren en structureren van het complete testproces. TMap is verdeeld in zes fases welke doorlopen zijn tijdens het afstudeerproject:

- Planning en beheer
- Voorbereiding
- Specificatie
- Uitvoering
- Afronding
- Inrichting en beheer infrastructuur.

De eerste fase is de planningsfase, waarbij de opdracht wordt vastgesteld samen met de testproducten en de testtechnieken. Naast een planning is een testrisico-analyse uitgevoerd, de testrisico-analyse beschrijft de risico's van verschillende systeemdelen of functionaliteiten wanneer deze niet naar behoren functioneren of misbruikt worden.
In de specificatiefase zijn testscripts geschreven voor de verschillende testobjecten. De testscripts beschrijven een herhaalbare testsituatie en de stappen die doorlopen dienen te worden om te kwaliteit van het testobject vast te stellen.
De uitvoeringsfase is de fase waarin het testteam de specificeerde tests uitvoeren op de testobjecten. Het testteam rapporteert bevindingen, opgedaan tijdens het uitvoeren van de tests, zodat deze bevindingen meegenomen kunnen worden in het testrapport.
Tijdens de afrondingsfase worden de bevindingen geanalyseerd en wordt het testobject geëvalueerd. De evaluatie wordt gedocumenteerd in een testrapport, waar ook een conclusie over de status van de testobjecten in beschreven wordt.
De laatste fase is de fase Inrichten en beheer infrastructuur. Binnen deze fase worden de verschillende testtechnieken en de testobjecten gespecificeerd.

5.4.2 OPSTELLEN TESTPLAN

Het opstellen van het testplan is gedaan door te kijken welke verschillende soorten testtechnieken in TMap relevant zijn voor het te realiseren systeem. Het testplan bevat onder meer informatie over het te testen systeem, de teststrategie en de testactiviteiten.

Centric test haar applicaties door middel van testscripts. De testscripts worden volgens een template in Excel opgesteld. In het template wordt uitleg gegeven over het te testen product en de situatie die getest wordt, het template kan meerdere testscripts bevatten. In het testscript wordt per stap de uit te voeren acties en het verwachte resultaat gedefinieerd.

5.4.3 TESTRISICO-ANALYSE

Voordat de daadwerkelijke tests zijn opgesteld, is een testrisico-analyse uitgevoerd. Tijdens de analyse is elke use case apart geanalyseerd en is een risiconiveau bepaald per use case. Het risiconiveau van een use case hangt af van hoe essentieel een use case voor het systeem is en welke toegang de use case heeft, zoals het muteren van data. Uit de analyse is gebleken dat vrijwel elke functionaliteit in beide applicaties een hoog risico heeft. De reden waardoor vrijwel elke functionaliteit een hoog risico heeft, is omdat gebruik is gemaakt van Agile Unified Process. AUP definieert dat alleen de essentiële elementen van een systeem geïmplementeerd worden, waardoor elke functionaliteit een essentiële functie heeft binnen het systeem en tevens een hoog risico vormt.

5.4.4 TESTSOORTEN

Tijdens het opstellen van het testplan zijn verschillende soorten tests bepaald. De verschillende testsoorten zijn gekozen in overleg met het testteam van Centric. In het testplan zijn zowel systeemtests, waaronder schermtests en consistentietests, als acceptatietests, zoals functionele acceptatietests en performancetests opgenomen.

Omdat tijdens de ontwikkeling gebruik is gemaakt van (TDD) is gekozen om vrijwel alleen whitebox tests toe te voegen aan het testplan. De opdrachtgever vindt het belangrijk dat eindgebruikers in de Viewer applicatie zonder lange laadtijden kunnen werken en dat de Centric stijlgids geïmplementeerd wordt.

Door de resultaten van de testrisico-analyse en de test-strategie is de nadruk gelegd op de belangrijke en kritieke delen van het systeem tijdens het testen.
De criteria voor de performancetests zijn bepaald in overleg met een consultant bij Centric.

5.4.5 UITVOEREN

Het uitvoeren van het testplan is gedaan door een tester van de afdeling. De tester heeft weinig kennis van de content van de opdracht, maar heeft genoeg verstand ervan om de functionaliteit van de applicaties te begrijpen. De tester heeft van mij de testscripts en het testplan gekregen, zodat hij weet wat uitgevoerd moet worden.

Wegens drukte bij het testteam zijn de performancetests door mij uitgevoerd. Het is mogelijk om dit te doen omdat het bij performancetests vrijwel alleen om tijdsmetingen gaat en niet om meningen.

6. CONSTRUCTION FASE

6.1 AGILE UNIFIED PROCESS BIJ CONSTRUCTION FASE

Tijdens de construction fase is elke Use case onafhankelijk van een ander ontworpen en geïmplementeerd. Gedeelde Use cases, zoals het exporteren naar een XML-bestand zijn zo ontworpen dat deze herbruikbaar zijn in beide applicaties.
Elke geïmplementeerde functionaliteit heeft eerst Unit-tests gekregen, zodat bij onderhoud vastgesteld kan worden waar en welke functionaliteit invloed heeft op andere functionaliteit.
Naast het implementeren van nieuwe functionaliteit zijn geregeld verbeteringen aangebracht op bestaande geïmplementeerde Use cases, Zoals verbeterde structuur of opdeling van functionaliteit om code duplicatie te voorkomen.

6.2 CENTRIC FRAMEWORK

Centric gebruikt een framework voor de applicaties die in .NET ontwikkeld worden. Het Centric framework zorgt ervoor dat de Centric huisstijl geimplementeerd wordt, volgens een bepaalde architectuur gewerkt wordt en dat de ontwikkelaar ondersteund wordt tijdens de ontwikkeling van de applicatie. Het Centric framework is ontwikkeld door medewerkers van Centric.
De opdrachtgever wenste dat de applicaties, die in dit afstudeerproject ontwikkeld worden, gebruik maken van het Centric framework.

Het Centric framework bevat, naast de hiervoor genoemde functionaliteit, een reeks componenten genaamd Centric controls. De Centric controls zijn formulier elementen die functionaliteit van componenten van DevExpress[6] uitbreiden en de Centric huisstijl toepassen.

Tijdens een korte introductie werd uitgelegd wat het Centric framework voor functionaliteit heeft en waarmee het de ontwikkelaar kan helpen. Tijdens de korte uitleg werd verteld dat een templateproject in ontwikkeling is. Het templateproject zou het voor ontwikkelaars eenvoudiger moeten maken om een nieuw project te starten.

6.2.1 DOCUMENTATIE

Er is geen documentatie van het Centric framework aanwezig. Op de lokale Sharepoint omgeving van Centric is wel documentatie van de Centric controls te vinden. Deze documentatie staat echter volledig los van het Centric framework.

[6] Externe .NET library.

6.2.2 EEN NIEUW PROJECT

Het starten van een nieuw project binnen het Centric framework werkte eenvoudig, echter merkte ik snel dat het framework veel functionaliteit overneemt van de ontwikkelaar. Onder andere mutatieformulieren en zoekformulieren zijn vrijwel volledig functioneel te genereren binnen het framework.
Na ongeveer vijf dagen met het framework te hebben gewerkt, bleek dat de huidige versie van het framework ongeschikt is voor applicaties die niet ten alle tijden met een database communiceert. Nadat overlegd is met de opdrachtgever, is besloten om het framework links te laten liggen. De opdrachtgever wenste wel dat het systeem gebruik maakt van Centric controls, dit zijn elementen om formulieren op te maken in de Centric huisstijl. Door het gebruik van Model View Presenter (MVP) binnen zowel het systeem als het Centric framework, is in de toekomst de functionaliteit zonder wijziging in structuur toe te passen in het Centric framework.

6.2.3 EVALUATIE

Het onder de knie krijgen van het Centric framework heeft zo'n vijf werkdagen gekost. In een volgende vergelijkbare situatie zou ik eerder om goede documentatie en voorbeelden vragen, wanneer deze niet beschikbaar zijn ga ik naar andere manieren zoeken om de informatie te verkrijgen die ik nodig heb om mijn doel te bereiken.

6.2.4 SUGGESTIES

Het Centric framework is opgesteld om ontwikkelaars te ondersteunen met het bouwen van applicaties waarbij voornamelijk data uit een database opgezocht, gemuteerd en toegevoegd kan worden. Het framework bevat wel basis classes waarmee de ontwikkelaar meer vrijheid wordt geboden, echter missen deze classes functionaliteit waardoor ze eigenlijk niet bruikbaar zijn.
Om het framework geschikt te maken voor elk type applicaties moet de ontwikkelaar meer vrij gelaten worden en moeten de basis classes meer functionaliteit bevatten.

De algemene mening over het Centric framework bij software ontwikkelaars op de MVV afdeling is niet positief. De ontwikkelaars spreken voornamelijk over een "ivoren toren" project, waarbij niet geluisterd wordt naar de echte ervaringen van de software ontwikkelaars die het framework gebruiken. Mijn suggestie om dit probleem op te lossen is het ontwikkelaarsteam die het framework ontwikkelt in korte periodes laten wisselen met andere ontwikkelaars, waarbij een hoofdontwikkelaar een langere periode actief blijft om ervoor te zorgen dat één architectuur behouden wordt binnen het framework.

6.4 ONTWIKKELEN VAN HET SYSTEEM

Het ontwikkelen van het systeem is gedaan door middel van het implementeren van de verschillende Use cases in volgorde zoals de gebruiker ze gaat doorlopen. De eerste applicatie die gebouwd is, is de Extractor. De Extractor wordt als eerste gebruikt om data te importeren vanuit een databron en om vervolgens data te exporteren naar een XML bestand.
De eerste Use case die geïmplementeerd is, is het importeren van data vanuit een Oracle database.

Nadat de Use cases van de Extractor geïmplementeerd waren, is de Viewer ontwikkeld. De Viewer is volgens hetzelfde principe als de Extractor ontwikkeld. De gebruiker moet als eerste een XML bestand importeren waarna de Viewer een databasediagram toont.

6.5 ONTWIKKELEN EXTRACTOR

6.5.1 OPHALEN DATABASESTRUCTUUR VANUIT ORACLE

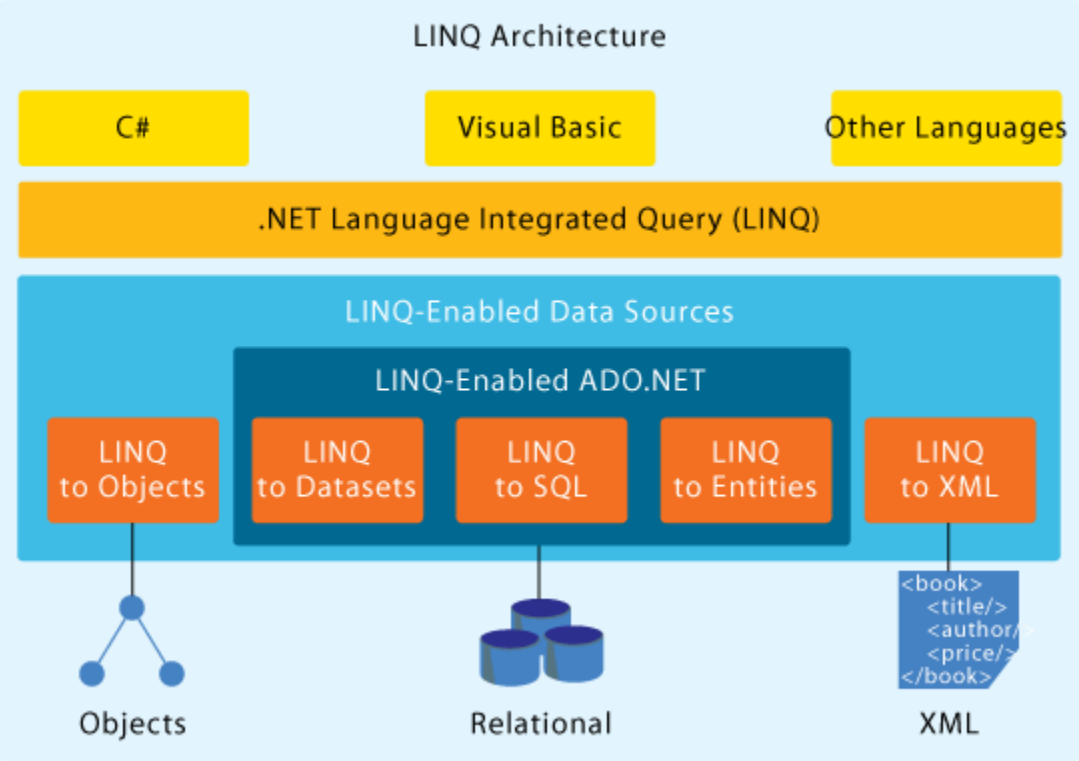
Om vanuit .NET te kunnen verbinden met een Oracle DBMS is het noodzakelijk om een Oracle client op de computer geïnstalleerd te hebben. Naast de officiele Client software van Oracle is het mogelijk om een applicatie van een derde partij hiervoor te gebruiken.
Centric maakt bij een aantal applicaties gebruik van de DotConnect data provider, wat mij overtuigde om DotConnect te gebruiken.
DotConnect heeft eigen methodes om metadata op te halen vanuit een Oracle database, waarmee geen eigenaar rechten nodig zijn. Doordat geen eigenaarrechten nodig zijn om de data op te halen, is het mogelijk voor mensen die geen database beheerder zijn om data uit te lezen. De opdrachtgever geeft de voorkeur aan het niet nodig hebben van eigenaar rechten binnen de Extractor applicatie.

De class die ervoor zorgt dat een export van metadata van Oracle databases gemaakt kan worden, staat volledig los van verdere logica van de Extractor. Er hoeft één enkele method aangeroepen te worden, welke de gehele import afhandelt. Door deze opzet van data importeren is één manier om het te gebruiken beschikbaar, waardoor onderhoud eenvoudig blijft.

6.5.2 TDD MET DOTCONNECT

DotConnect wordt in één use case gebruikt om ervoor te zorgen dat data vanuit een Oracle database geïmporteerd kan worden. De use case waarbij DotConnect gebruikt wordt is helaas niet volledig met unit-tests te testen. DotConnect ondersteunt geen vorm van gestructureerd testen en is daardoor onmogelijk in een onafhankelijke situatie te testen.

6.5.2 LANGUAGE INTEGRATED QUERY



Afbeelding 10, Schematische weergave van de architectuur van LINQ.[7]

Tijdens het ontwikkelen van de Extractor is gebleken dat het importeren van data vanuit een Oracle database lang duurde. Tijdens een meting is geconstateerd dat het ophalen van vreemde sleutels de oorzaak van de lange wachttijden was. In Oracle worden vreemde sleutels via constraintnamen aan primaire sleutels gekoppeld. De gegevens van de constraints staan in de user_constraints tabel en de user_cons_columns tabel. DotConnect heeft enkel de mogelijkheid om de constraints van vreemde sleutels, primaire sleutels en de koppeling tussen deze constraints elk apart op te halen, waardoor drie SQL query's nodig zijn om de gegevens op te halen.
bij het verwerken van vreemde sleutels moeten de resultaten van de drie SQL query's gecombineerd worden om de vreemde sleutels met de juiste primaire sleutels te koppelen. Om de gewenste gegevens op te halen wordt gebruik gemaakt van de GetSchema functionaliteit van DotConnect. Doordat de GetSchema functionaliteit gebruikt wordt, zijn geen eigenaar rechten nodig om alle gegevens op te halen.

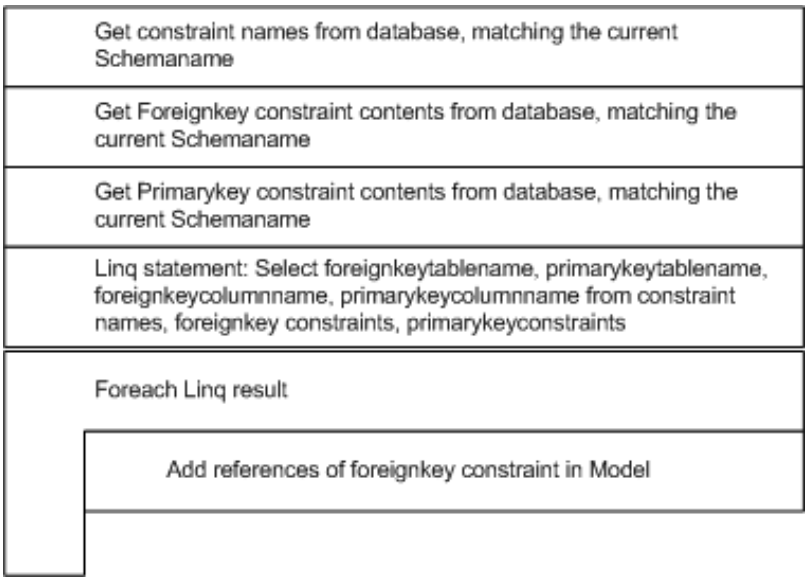
Om de snelheid van bovengenoemde functionaliteit te verhogen ben ik op zoek gegaan naar technieken waarmee collecties van data met elkaar gecombineerd kunnen worden om een resultaat op te leveren. Om te zoeken naar een nieuwe techniek om mijn probleem op te lossen heb ik het boek “Pro C# with .NET 3.0” door Andrew Troelsen gebruikt, hierin stond informatie over een techniek genaamd Language Integrated Query binnen .NET waarmee verschillende collecties van data gecombineerd kunnen worden, net als een join in een SQL query.

[7] Bron:http://www.codeproject.com

Language integrated query (LINQ) is een techniek waarmee datacollecties kunnen worden gecombineerd om een resultaat te vinden. Een aantal databronnen waarmee LINQ kan werken zijn databases, .NET objecten en XML. LINQ wordt in deze situatie gebruikt om de resultaten van de drie SQL query's te combineren en heeft een prestatie winst van zo'n 700% opgeleverd. Onderstaand tabel toont de snelheidswinst die het gebruik van LINQ heeft opgeleverd.

Actie	Tijdsduur tijdens ophalen testdatabase zonder LINQ	Tijdsduur tijdens ophalen testdatabase met LINQ
Ophalen van tabellen	3 seconden	2 seconden
Ophalen van views	7 seconden	7 seconden
Ophalen van functions	8 seconden	9 seconden
Ophalen van kolommen	14 seconden	12 seconden
Ophalen van primaire sleutels	4 seconden	4 seconden
Ophalen van vreemde sleutels	1004 seconden	120 seconden
Totaal	1040 seconden	154 seconden

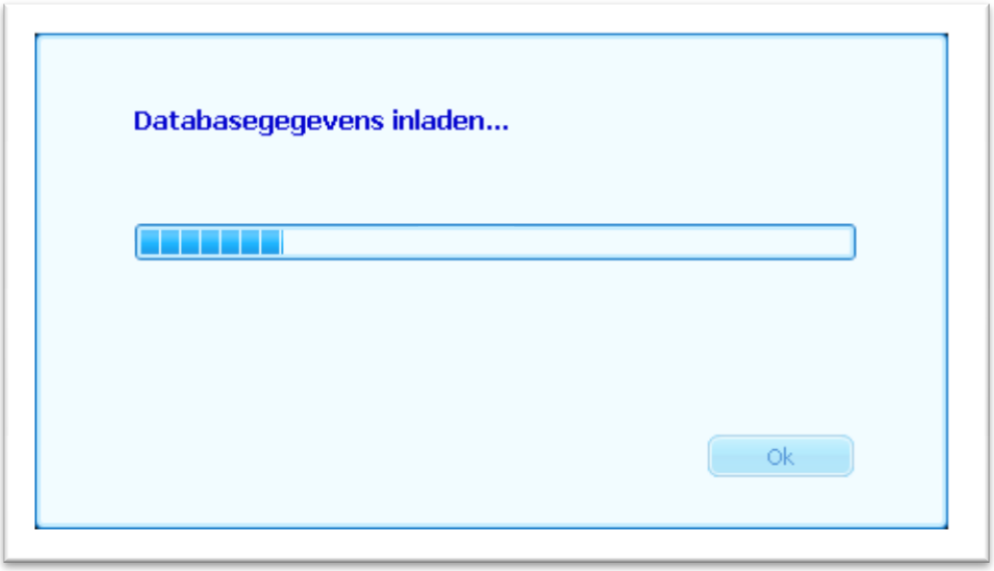
LINQ heeft naast betere performance ook voor een overzichtelijkere methode gezorgd. In plaats van het doorlopen van meerdere collecties van data is nu één LINQ statement nodig om alle resultaten bij elkaar te zoeken. Onderstaand diagram (afbeelding 11) toont het Programma Structuurdiagram (PSD) van de methode waarmee de vreemde sleutels opgehaald worden. Door het toepassen van LINQ is van de vijf foreach iteraties in de oude situatie slechts één overgebleven.



Afbeelding 11, PSD van het ophalen van vreemde sleutels vanuit een Oracle DBMS

6.5.3 VOORTGANGSINDICATOR BIJ OPHALEN ORACLE GEGEVENS

Het ophalen van data uit een Oracle DBMS duurt met de testdatabase van Centric zo'n 154 seconden. Om ervoor te zorgen dat de gebruiker een idee heeft dat de Extractor bezig is met het verwerken van het verzoek van de gebruiker is een voortgangsindicator toegevoegd. De voortgangsindicator is een element in de View, waarmee de voortgang van een proces getoond kan worden.



Afbeelding 12, Voortgangsindicator van de Extractor

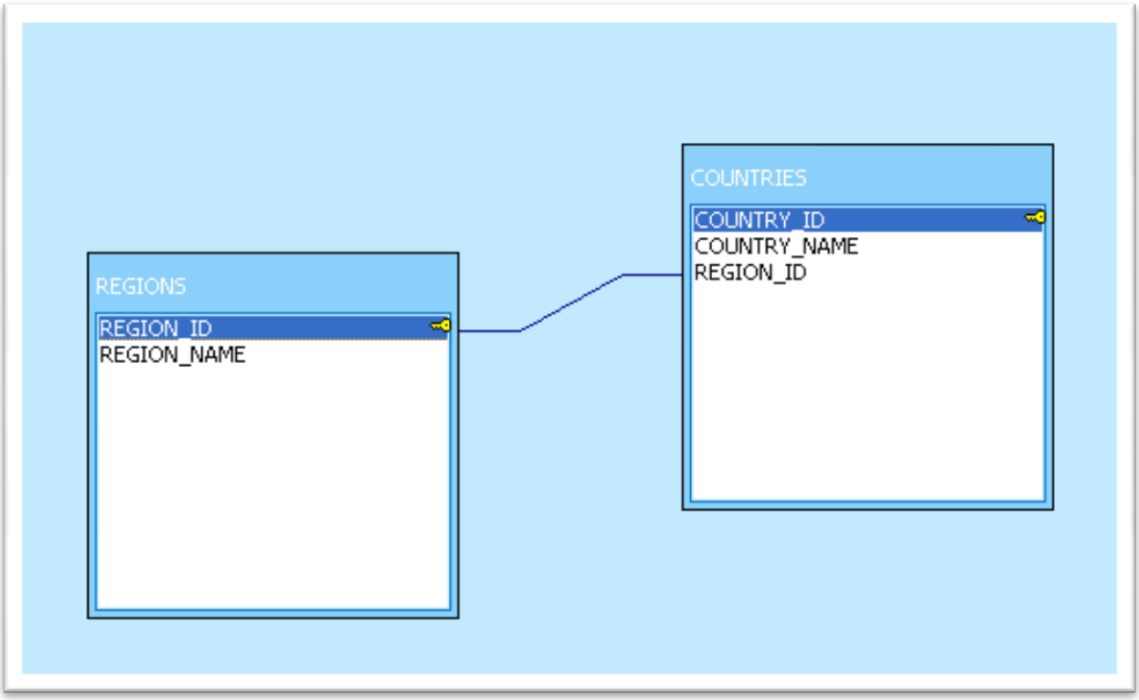
De voortgangsindicator bij het ophalen van gegevens uit een Oracle DBMS toont geen geleidelijke voortgang, maar springt in één keer naar 100 procent wanneer het proces klaar is. De reden van de incorrecte voortgangsindicatie is doordat de class waarmee de gegevens opgehaald worden maar één methode heeft waarmee het hele proces gestart wordt, waardoor de Presenter in de tussentijd geen feedback kan geven aan de View.

Een oplossing voor bovenstaand probleem is het aanmaken van een methode in de BLL die dient als doorgeefluik naar de voortgangsindicator in de gebruikersinterface. De Oracle database importeer class moet tijdens het aanmaken van een instantie een referentie meekrijgen naar de bovengenoemde methode, deze referentie moet via een MethodInvoker instantie gaan. De Oracle database importeer class heeft door bovenstaande oplossing geen koppeling met de BLL of de Presenter, waardoor de functionaliteit nog steeds herbruikbaar is, maar wel de mogelijkheid biedt tot feedback naar de View.

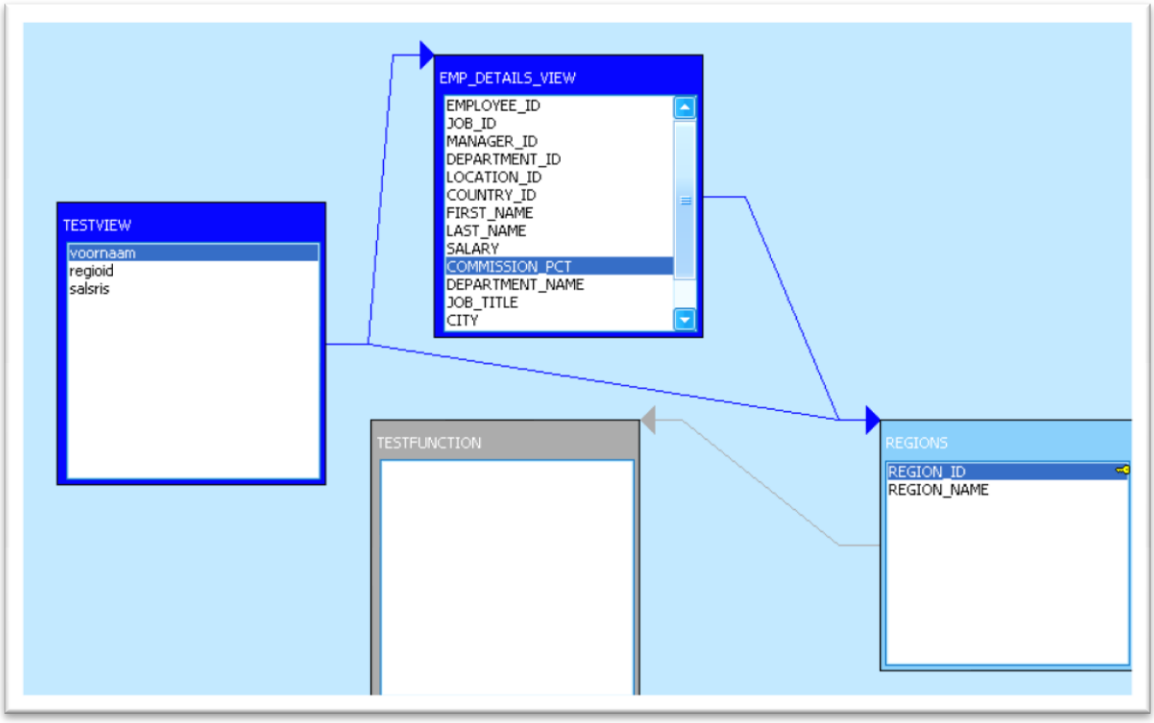
6.6 ONTWIKKELEN VIEWER

6.6.1 GENEREREN DATABASEDIAGRAM

De belangrijkste functionaliteit van de Viewer is het tonen van een databasediagram. Tijdens het ontwikkelen van de basisfunctionaliteit, is het tonen van een diagram van het volledig ingeladen XML-bestand geïmplementeerd. Het diagram bestaat uit een aantal verschillende Usercontrols met elementen waarin de data weergegeven wordt. In .NET is de UserControl class een basiselement die gebruikt kan worden om elementen voor de gebruikersinterface te maken. De Viewer is tevens in staat om relaties tussen verschillende elementen te tonen. De relaties worden getoond door middel van lijnen die tussen de elementen getekend worden. Zie onderstaande illustratie.



Afbeelding 13, Een relatie tussen twee tabellen op basis van een vreemde sleutel. Primaire sleutels worden aangegeven met een sleutel icoon.



Afbeelding 14, Relaties tussen tabellen, views en functions. De pijlen wijzen altijd naar de bovenkant van het doelelement.

De geïmplementeerde Usercontrols zijn verplaatsbaar door ze met de muis te slepen.
Wanneer het volledige diagram getoond wordt en de gebruiker een element verplaatst, wordt de locatie van het verplaatste element opgeslagen in het Model. Wanneer de XML exporteer functionaliteit in de Viewer gebruikt wordt, wordt de locatie van alle elementen opgeslagen, zodat een gebruiker maar één keer het diagram in hoeft te delen.

6.7 ONTWIKKELEN GEDEELDE FUNCTIONALITEIT

6.7.1 MODEL

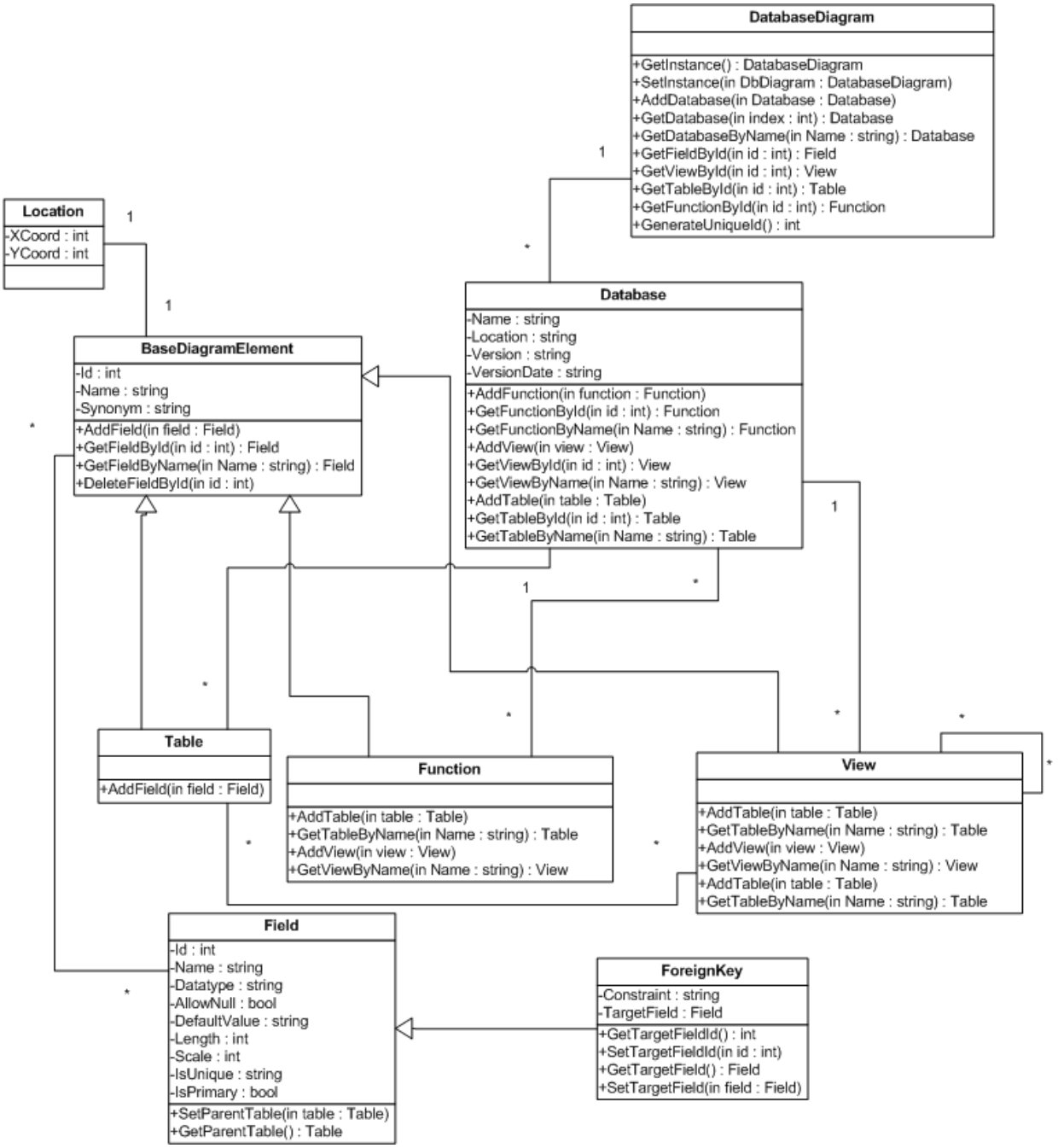
Het Model is in beide applicaties gelijk. Deze classes zijn los van de rest van de applicaties ontwikkeld en bevatten enkel functionaliteit die herbruikbaar is in meerdere situaties.

Onderstaande afbeelding geeft het Model van het systeem weer. Het hoofdelement is de DatabaseDiagram class, deze class is een Singleton class en is een Container voor instanties van de Database class. Wanneer een class een Container is, betekend het dat deze class enkel bedoeld is om een collectie van andere classes in op te slaan.
De Database class bevat velden waar onder andere de naam en locatie van een specifieke database in opgeslagen kan worden.
Het systeem ondersteund momenteel enkel drie verschillende database-elementen, de tabel, de view en de functie. Elk element binnen een database erft functionaliteit over van de BaseDiagramElement class en kan referenties hebben naar andere implementaties van de BaseDiagramElement class. De BaseDiagramElement Biedt tevens de mogelijkheid om een locatie op te slaan, de locatie dient ervoor om de elementen op het scherm te tonen in de Viewer. De Field class representeert onder andere een kolom binnen een tabel of een parameter van een functie.

De classes BaseDiagramElement en Field bevatten een attribuut waarin een uniek Id wordt opgeslagen. Het unieke Id wordt gegenereerd tijdens het importeren vanuit Oracle en zorgt ervoor dat een unieke referentie naar objecten binnen het systeem mogelijk is.

Het diagram is ontworpen aan de hand van de structuur waarop de verschillende elementen in Oracle databases opgeslagen worden. Door middel van de PL/SQL Developer[8] applicatie is gekeken hoe relaties tussen elementen binnen Oracle geregeld worden. Het Model ondersteund relaties niet op dezelfde manier als in Oracle, maar is een implementatie die voor meerdere verschillende DBMS-en toepasbaar is.

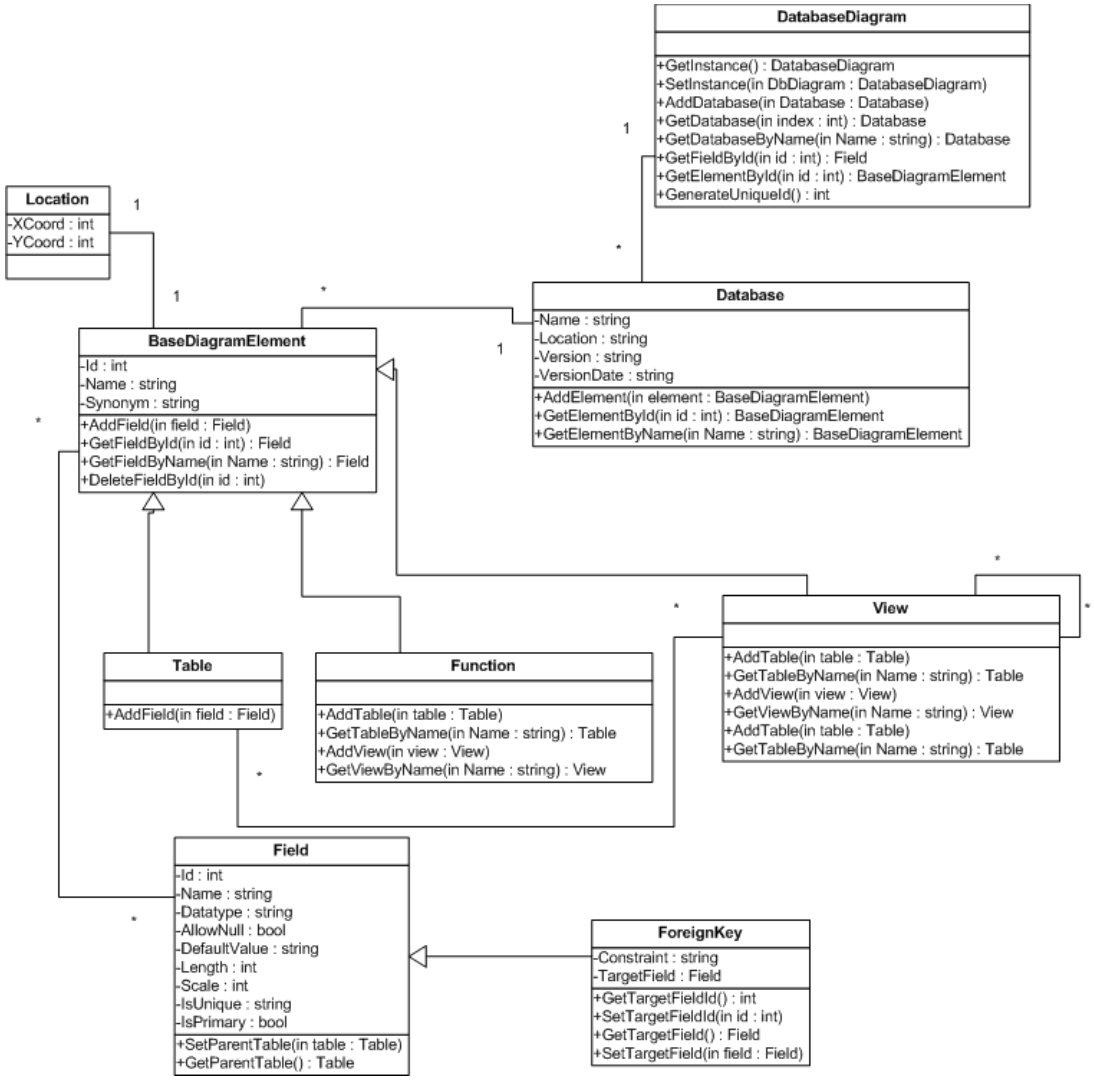
[8] Applicatie ontwikkeld door AllroundAutomatations, waarmee Oracle databases aangepast kunnen worden.



Afbeelding 15, Classdiagram van het Model van beide applicaties.

VERBETERINGEN IN MODEL

Het Model bevat een fout. De fout is geconstateerd tijdens het opstellen van het construction rapport. De fout heeft geen functionele belemmering als gevolg, maar maakt toekomstig onderhoud lastiger. De fout is dat de Database class niet direct een relatie met de BaseDiagramElement class heeft, maar enkel bij de overervende classes. In de situatie dat de Database class een directe relatie met de BaseDiagramElement heeft, wordt het toevoegen van elementen bij toekomstig onderhoud eenvoudiger.
De verbetering in het Model zorgen voor het volgende Model.

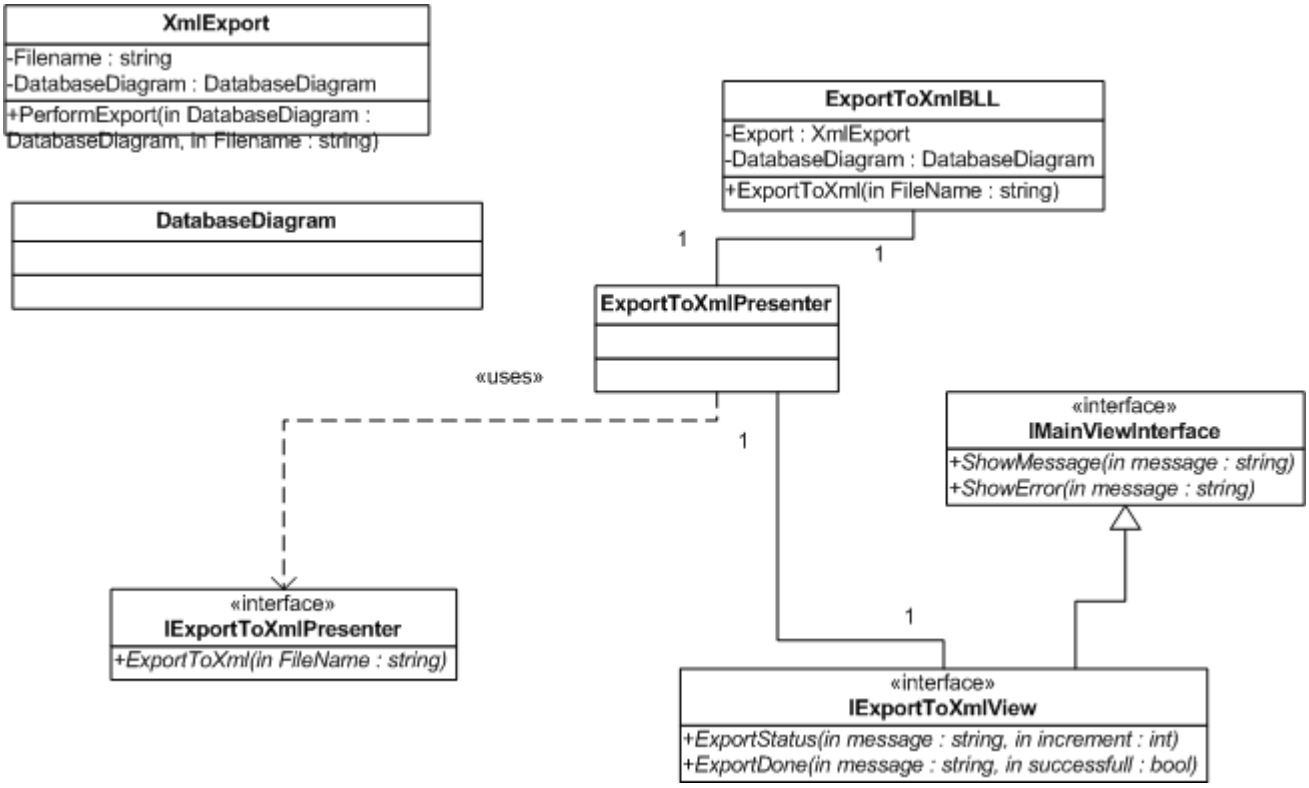


Afbeelding 16, Model na doorvoeren verbeteringen.

6.7.2 GEDEELDE MVP'S

Om het mogelijk te maken dat beide applicaties in sommige gevallen gebruik maken van dezelfde MVP instanties is een gedeelde module toegevoegd. De gedeelde module bevat de Presenter, Presenter interface, View interface en BLL die nodig zijn om de gedeelde functionaliteit te kunnen implementeren. Er is voor gekozen om de View implementaties niet in deze module toe te voegen omdat hierdoor de functionaliteit minder herbruikbaar is. Het moet immers mogelijk zijn om een webinterface of commandline-interface toe te voegen als alternatief op de geïmplementeerde windows-forms interface.

Beide applicaties bevatten de functionaliteit om XML bestanden te kunnen importeren en exporteren. Onderstaand diagram toont een klassendiagram van het exporteren van data naar een XML-bestand en toont de structuur van een geïmplementeerde use case binnen het gedeelde deel van het systeem. Zoals in onderstaand diagram te zien is, wordt de View-implementatie niet in het gedeelde deel van het systeem opgenomen.



Afbeelding 17, Klassendiagram van het exporteren van data naar een XML-bestand.

6.7.3 XML PARSEN

Beide applicaties maken gebruik van een XML-databron. Zowel de Extractor als de Viewer hebben de functionaliteit om data naar een XML-bestand te schrijven. Het importeren van een XML-bestand is op dit moment alleen in de Viewer geïmplementeerd. De logica en de MVP classes om toegang te krijgen tot een XML bestand zijn opgenomen in de gedeelde module.

```
    <Function Id="1334831936" Name="WWV_POPUP_FILTER">
      <Location XCoord="0" YCoord="0" />
    </Function>
  </Database>
  <Database Id="1022777938" Location="127.0.0.1" Name="HR">
    <Table Id="1748255403" Name="COUNTRIES">
      <Field Id="682424062" Name="COUNTRY_ID" DataType="CHAR" AllowNull="false" DefaultValue="" Len
      <Field Id="673392022" Name="COUNTRY_NAME" DataType="VARCHAR2" AllowNull="true" DefaultValue="
      <Field xsi:type="ForeignKey" Id="1226352090" TargetFieldId="1444240007" Name="REGION_ID" Date
      <Location XCoord="0" YCoord="0" />
    </Table>
    <Table Id="1365198196" Name="DEPARTMENTS">
      <Field Id="1469700646" Name="DEPARTMENT_ID" DataType="NUMBER" AllowNull="false" DefaultValue=
      <Field Id="1477109066" Name="DEPARTMENT_NAME" DataType="VARCHAR2" AllowNull="false" DefaultVa
      <Field xsi:type="ForeignKey" Id="1143874855" TargetFieldId="1823076290" Name="LOCATION_ID" Da
      <Field xsi:type="ForeignKey" Id="1817557116" TargetFieldId="1057109678" Name="MANAGER_ID" Dat
      <Location XCoord="0" YCoord="0" />
    </Table>
    <Table Id="503230149" Name="EMPLOYEES">
      <Field Id="1057109678" Name="EMPLOYEE_ID" DataType="NUMBER" AllowNull="false" DefaultValue=""
```

Afbeelding 18, Deel van een XML-bestand, geëxporteerd door de Extractor.

EXPORTEREN NAAR XML

De exporteer functionaliteit naar XML van de Extractor maakt gebruik van de XmlSerializer class van .NET. De XmlSerializer class maakt een XML representatie van het opgegeven object en schrijft deze weg naar een XML bestand. De XmlSerializer houdt rekening met collecties van andere objecten die in het opgegeven object zitten en exporteert deze recursief naar het XML bestand. De classes in het Model bevatten referenties naar elkaar, iets wat niet ondersteund wordt door de XmlSerializer. Om die reden is de XML-exporteer class uitgebreid met methodes die referenties naar objecten weghalen en vervangt met referentie op Id niveau.

Net als de Oracle database import class, heeft de XML-exporteer class één methode welke de totale export regelt.

IMPORTEREN VANUIT XML

Net als de XML-exporteer functionaliteit, maakt de XML-importeer functionaliteit gebruik van de XmlSerializer class van .NET. Omdat de geimporteerde objecten geen objectreferentie naar elkaar hebben, bevat de XML-importeer class methodes om Id-referentie om te zetten naar objectreferentie. De XML-importeer class heeft één publieke methode waarmee de gehele importeer functionaliteit uitgevoerd wordt.

6.8 TESTRESULTATEN

De testresultaten en de bevindingen van de tests zijn door de tester genoteerd en later door mijzelf in het testrapport verwerkt samen met de performancetest resultaten.
Uit het testrapport blijkt dat beide applicaties nog niet gereed zijn voor een gebruik, onderstaande tabel toont het aantal bevindingen per applicatie.

Applicatie	Erg hoog	Hoog	Midden	Laag	Totaal
Extractor	0	0	1	2	3
Viewer	1	0	2	3	6
Totaal	1	0	3	5	9

De bevindingen met prioriteit "Midden" en "Laag" zijn bevindingen waardoor de werking van de applicaties niet geblokkeerd wordt, terwijl de bevinding met "Erg hoog" een ernstige bevinding is die een belangrijke functionaliteit van een applicatie blokkeert.

BEVINDINGEN

Tijdens het uitvoeren van de verschillende tests zijn een aantal bevindingen opgedaan. Elke bevinding heeft een prioriteit meegekregen aan de hand van de impact van de bevinding. Één bevinding heeft een erg hoge prioriteit omdat deze bevinding ervoor zorgt dat de applicatie stopt met functioneren wanneer een heel groot databasediagram getoond moet worden.

De bevinding met erg hoge impact wordt veroorzaakt door een limiet in de Windows.Forms controls van .NET, deze controls kunnen enkel 10.000 elementen onder zich hebben. Wanneer een groot databasediagram, zoals de TestMV van Centric, ingeladen wordt, wordt de limiet van 10.000 elementen overschreden.
Dit probleem is met de opdrachtgever besproken en er is een manier bedacht om deze bevinding van minder belang te laten worden. Volgens de opdrachtgever is een diagram van het formaat van de TestMV database vrijwel nutteloos om in complete vorm in te zien en om die reden is een manier bedacht om de informatie overzichtelijker aan de gebruiker aan te bieden. Door de bedachte manier om informatie overzichtelijker aan te bieden is de prioriteit van de bevinding een stuk lager geworden en volstaat een melding wanneer aan dit limiet voldaan is.
De bevinding is enkel een probleem wanneer gebruik gemaakt wordt van de View met Windows.Forms controls.
Op het moment dat de opdrachtgever besluit om een webinterface te implementeren voor de Viewer, is het mogelijk dat de bevinding voor de nieuwe interface opgelost is.

De overige bevindingen zijn allemaal bevindingen met weinig tot vrijwel geen impact.

7. ITERATIE: OVERZICHTELIJKHEID

7.1 AANLEIDING

Tijdens het realiseren van de functionaliteit van de Viewer applicatie, is gekeken naar de overzichtelijkheid van de gegenereerde diagrammen. Al snel bleek dat wanneer een diagram van meer dan 20 elementen getoond werd, het diagram onoverzichtelijk werd.
Tijdens een overleg met de opdrachtgever heb ik voorgesteld om een iteratie te ontwerpen en ontwikkelen waarmee de Viewer applicatie gebruiksvriendelijker wordt. De opdrachtgever was akkoord gegaan met de te realiseren functionaliteit die geïmplementeerd wordt in deze iteratie.

INCEPTION FASE

Tijdens de inception fase zijn keuzes gemaakt hoe de gebruiker gebruik gaat maken van de te realiseren functionaliteit. Er is voor gekozen om twee manieren om diagram elementen aan een database diagram toe te voegen te implementeren.

- Toevoegen door middel van slepen vanuit een lijst van elementen.
- Toevoegen door middel van klikken op een element in een lijst van elementen.

Beide manieren hebben een eigen functionaliteit, de eerste voegt enkel het gesleepte element aan het getoonde database diagram toe. De tweede manier voegt het geselecteerde element en de elementen waar het geselecteerde element naartoe/van verwijzingen heeft aan het getoonde database diagram toe.

7.2 ELABORATION FASE

Tijdens de elaboration fase van de "overzichtelijkheid" iteratie heeft het ontwerp van de Use case waarbij een compleet database diagram getoond wordt gewijzigd. De te realiseren functionaliteit en de reeds geïmplementeerde functionaliteit in de Viewer applicatie bevatten overeenkomsten, waardoor code duplicatie zou ontstaan wanneer de bestaande functionaliteit ongewijzigd zou blijven.
Er is besloten om een basisimplementatie te ontwikkelen voor het tonen van database diagrammen in de Viewer applicatie. De te realiseren functionaliteit overerft functionaliteit van de basisimplementatie, net als de implementatie van de functionaliteit waarbij het complete diagram getoond wordt. Door de wijziging in de bestaande code in de Viewer is het in de toekomst eenvoudiger om nieuwe functionaliteit toe te voegen zonder dat code duplicatie ontstaat.

7.3 ONTWIKKELING

De ontwikkeling van deze iteratie is door tijdgebrek helaas niet doorgegaan. De keuze om de implementatie buiten de afstudeerperiode uit te voeren, is genomen in overleg met de opdrachtgever. Het ontwerp en de requirements zijn wel vastgesteld, maar nog niet geïmplementeerd.

8. EVALUATIE

8.1 PROCESEVALUATIE

8.1.1 AANVANG VAN PROJECT

Bij aanvang van het afstudeerproject was het niet mogelijk om direct bij Centric op kantoor te beginnen. Centric heeft een regelgeving waarin staat dat slechts op één moment van de maand werknemers en stagiaires aangenomen kunnen worden. Het moment waarop werknemers aangenomen konden worden in februari was op 22 februari.
Door bovenstaande regelgeving heb ik twee weken thuis gewerkt aan het project. Communicatie met de opdrachtgever ging op afspraak en ik heb in de eerste twee weken aan het plan van aanpak en aan de requirements gewerkt. Deze situatie heeft wel enige vertraging opgeleverd omdat communicatie met de opdrachtgever indirecter was dan wanneer ik op kantoor aanwezig zou zijn.

8.1.2 AFHANKELIJKHEID

Wanneer een project met minder prioriteit afhankelijk is van het werk van een collega, is het mogelijk dat een situatie voordoet waarbij vertraging ontstaat. Aangezien mijn afstudeerproject minder prioriteit heeft dan het oplossen van incidenten, ontstond af en toe een situatie waarbij een deel van het project niet verder kon.
Tijdens het project heb ik meerdere malen een situatie gehad waarbij ik afhankelijk was van de opdrachtgever of van een collega.

De eerste situatie was ontstaan door het later ingaan van mijn afstudeercontract. Tijdens de eerste twee weken heb ik thuis gewerkt aan de definitie van het project, waarbij communicatie met de opdrachtgever cruciaal is. De opdrachtgever antwoordde soms niet direct op de vragen die ik gestuurd had, waardoor ik niet verder kon met het definiëren van de requirements. In de tijd dat er op antwoord van de opdrachtgever gewacht moest worden, is aan zelfstudie gedaan om meer te weten te komen over .NET en Oracle.

Een tweede situatie deed zich voor bij het reviewen van de applicaties door een collega. De collega die aangewezen was om mijn werk te reviewen had het regelmatig druk met het oplossen van incidenten, waardoor het reviewen werd uitgesteld. Door deze situatie is geen vertraging ontstaan, maar is de kwaliteit van de opgeleverde applicaties iets minder bevestigd.

Bij een volgend project is het wellicht beter om ervoor te zorgen dat ten alle tijden een alternatieve taak ligt waarmee verder gewerkt kan worden wanneer gewacht moet worden op informatie of de werkzaamheden van een collega of opdrachtgever.

8.1.3 AGILE UNIFIED PROCESS

De keuze om Agile Unified Process (AUP) te gebruiken tijdens het afstudeerproject kwam tot stand doordat de opdrachtgever wenste dat het project volgens een Agile methode uitgevoerd werd, waarbij ik de specifieke Agile methode uit mocht zoeken. Door de combinatie van Agile technieken en technieken uit Rational Unified Process (RUP), is ervoor gekozen om Agile Unified Process voor mijn afstudeerproject te gebruiken.

Het ontwikkelen volgens de AUP methode vond ik lastig. Doordat ik vanuit de HHS enkel ervaring met RUP heb en tijdens de stage in het derde jaar een compleet andere methode genaamd "Getting Real" gebruikt heb, is het voor mij lastig om beide te combineren in één methode. Tijdens het project heb ik een aantal keer de neiging gehad meer richting de RUP kant op te gaan, omdat dit de meest bekende manier van software ontwikkelen voor mij is. Bij een volgend project zou ik een duidelijkere methode gaan gebruiken in plaats van een mix tussen twee methodes zoals AUP.

8.1.4 TEST DRIVEN DEVELOPMENT

Tijdens het project is ontwikkeld volgens Test Driven Development (TDD). Binnen AUP wordt TDD gezien als een goede manier om een systeem te testen en te ontwikkelen. Het gebruik van TDD beviel mij zeer goed. Doordat eerst unit-tests geschreven worden, is het niet mogelijk om extra onnodige methodes toe te voegen aan het systeem, waardoor het systeem eenvoudig blijft. Tijdens het onderhoud van een systeem is TDD handig, omdat het test of een wijziging impact op andere functionaliteit heeft gehad, waardoor toekomstige incidenten voorkomen kunnen worden. In een volgend project ga ik TDD weer gebruiken.

8.1.4 CENTRIC FRAMEWORK

In mijn afstudeerplan staan een aantal risicofactoren gedefinieerd. Uiteindelijk waren de risicofactoren die in mijn afstudeerplan staan, niet volledig. Na een aantal dagen met het Centric framework gewerkt te hebben bleek dat dit framework ongeschikt is voor het systeem dat door mij gerealiseerd is. Door bovenstaand probleem was het niet mogelijk om het Centric framework te gebruiken voor het te realiseren systeem. Om het systeem toch de huisstijl van Centric mee te geven, is gebruik gemaakt van de Centric Controls. De Centric Controls zijn gebruikersinterface elementen die overerven van de DevExpress[9], waar de Centric huisstijl al op toegepast is.
Voorafgaand aan een volgend project zou ik externe frameworks of library's als risicofactor opnemen, omdat dit onbekende en soms ongedocumenteerde factoren zijn.

[9] Externe .NET usercontrol library, ontwikkeld door Developer Express.

8.1.5 MODEL VIEW PRESENTER

Tijdens het ontwikkelen van het systeem heb ik de voordelen van MVP gerealiseerd. Het scheiden van functionaliteit op basis van use cases vind ik een logische aanpak en zorgt ervoor dat functionaliteit logisch gegroepeerd blijft.
Tevens het scheiden van de gebruikersinterface met de achterliggende logica is praktisch omdat verschillende typen gebruikersinterface ontwikkeld kunnen worden zonder logica te dupliceren.
In een volgend project ga ik MVP zeker overwegen om te gebruiken wanneer dat mogelijk is en bij bestaande software past.

8.1.6 BEDRIJF

De ervaring met Centric IT Solutions is over het algemeen goed. Op de afdeling hangt een rustige sfeer en werken voornamelijk vriendelijke mensen die, wanneer mogelijk, tijd nemen om een ander te helpen. Af en toe was het lastig om een afspraak te maken met een collega, omdat klantgerelateerde zaken meer prioriteit hebben in vergelijking met mijn project. Wekelijks vond een informele bespreking met de opdrachtgever plaats over de voortgang, knelpunten en wijzigingen in het project.

Een volgend project bij Centric zou ik zeker overwegen, maar bij afhankelijkheid van andere mensen bij Centric moet rekening worden gehouden met de beperkte beschikbare tijd van collega's.

8.2 PRODUCTEVALUATIE

8.2.1 MODEL INTERFACE

In de huidige architectuur wordt het Model direct aangesproken. Om ervoor te zorgen dat de Presenter en BLL met verschillende Model implementaties bruikbaar zijn, is het noodzakelijk om voor elke class in het Model een interface aan te maken.
Wanneer gebruik gemaakt wordt van interfaces heeft de Presenter geen weet van de implementatie van het Model en ontstaat Loose coupling. Loose coupling is een begrip in software architectuur en staat voor de mate waarin de ene class kennis heeft van een andere class.

8.2.2 DOTCONNECT

Het gebruik van DotConnect in de Extractor heeft ervoor gezorgd dat de ontwikkeling van de Extractor een stuk sneller is gegaan. Persoonlijk vind ik één aspect van DotConnect slecht, namelijk dat DotConnect geen ondersteuning voor unit-tests en mockdatabases biedt waardoor de functionaliteit niet door middel van unit-tests testbaar is.

8.2.3 SYSTEEM ALGEMEEN

Het systeem is helaas niet ontwikkeld in combinatie met het Centric framework, maar implementeert de Centric stijlgids wel. Het systeem is volgens dezelfde architectuur als het Centric framework ontwikkeld, zodat het systeem in de toekomst zonder teveel wijzigingen om te zetten is naar het framework. De architectuur zorgt er tevens voor dat verschillende types gebruikersinterface implementeerbaar zijn, zonder de logica te dupliceren.

Conclusie
Het systeem is erg modulair opgezet met veel mogelijkheid voor hergebruik van logica en het uitbreiden van het systeem.

8.2.3 ALGEMEEN EXTRACTOR

De Extractor applicatie bevat de basisfunctionaliteit zoals in de requirements vastgesteld is. De Extractor is, naar mijn mening, een goed onderhoudbare applicatie. De geïmplementeerde functionaliteit is een goede basis voor toekomstige functionaliteit en bevat de Centric huisstijl.

Conclusie
De Extractor is een simplistische applicatie waarmee Centric op een eenvoudige manier de structuur van databases kan importeren, waarbij geen technische kennis nodig is.

8.2.4 ALGEMEEN VIEWER

De Viewer ondersteunt de basisfunctionaliteit zoals in de requirements vastgelegd is. Helaas is wegens tijdgebrek geen extra functionaliteit toegevoegd die de Viewer bruikbaarder maakt.
De Viewer kan omgaan met grote hoeveelheid data zonder dat de gebruiker een lange tijd moet wachten tijdens het werken met de data.

Conclusie
De Viewer bevat een goede basis voor toekomstige functionaliteit, maar mist functionaliteit van de "overzichtelijkheid" iteratie waardoor de Viewer een echt bruikbare applicatie zou worden.

8.2.5 DOCUMENTATIE

De documentatie van het systeem is voldoende bevonden door de opdrachtgever, het testteam en een ontwikkelaar bij Centric. De documentatie is volgens het AUP principe “Just barely good enough” geschreven, waardoor sommige documenten wat kort lijken. Zoals in hoofdstuk 8.1.3 is aangegeven zou ik in een volgend project AUP niet graag gebruiken en het principe “Just barely good enough” vond ik een lastige manier van documenteren.

9. VERANTWOORDING BEROEPSTAKEN

9.1 UITVOEREN ANALYSE DOOR DEFINITIE VAN REQUIREMENTS

Volgens het afstudeerplan wordt deze beroepstaak op niveau 4 uitgevoerd. Tijdens het afstudeerproject is deze beroepstaak niet op het vooraf gedefinieerde niveau uitgevoerd. De beroepstaak "Uitvoeren analyse door definitie van requirements" is uitgevoerd op niveau 2. Het systeem omvat een beperkt aantal requirements en heeft een beperkt aantal stakeholders. Deze beroepstaak is zelfstandig uitgevoerd met een simpele complexiteit.
De reden van het complexiteitverschil is voornamelijk gekomen doordat met de opdrachtgever besproken is dat tijdens het ontwikkelen van het systeem voornamelijk op de belangrijkste functionaliteit geconcentreerd wordt, waardoor minder requirements zijn gedefinieerd en meer aandacht aan de architectuur besteed kan worden.
Tijdens het afstudeerproject is door mij geconstateerd dat een uitbreiding noodzakelijk is voor de bruikbaarheid van het systeem. De uitbreiding is beschreven in hoofdstuk 7 en is in bespreking met de opdrachtgever tot stand gekomen.

9.2 ONTWERPEN SOFTWAREARCHITECTUUR

In het afstudeerplan staat dat de beroepstaak "Ontwerpen softwarearchitectuur" op het niveau 3 behaald wordt tijdens het afstudeerproject.
Deze beroepstaak is tijdens het afstudeerproject op niveau 3 behaald door het bepalen en ontwerpen van de architectuur van het complete systeem. De beroepstaak is zelfstandig uitgevoerd, waarbij de complexiteit van de opdracht lastig was.
De herbruikbaarheid van de architectuur heeft zich bewezen doordat verschillende functionaliteit zonder enige aanpassing hergebruikt kan worden in meerdere situaties in meerdere gebruikersinterfaces en zelfs in meerdere applicaties. De kwaliteit van de architectuur is bevestigd door de softwarearchitect werkzaam bij de afdeling waar ik werkzaam ben.
Het gebruik van Model View Presenter (MVP) heeft ertoe geleid dat de ontwikkeling van elke use case van het systeem onafhankelijk uitgevoerd kon worden, waardoor een vorm van Feature Driven Development (FDD) is ontstaan.
De implementatie van Data Access Layers heeft ertoe geleid dat importeer- en exporteerfunctionaliteit in meerdere use cases bruikbaar zijn.

Door de gebruikte softwarearchitectuur is het voor Centric mogelijk om delen van het gerealiseerde systeem te gebruiken in haar huidige software of eventueel beide applicaties samen te voegen tot één applicatie.

Het ontwerpen van de architectuur is gedaan door middel van UML diagrammen, waarmee de objectoriëntatie binnen het systeem vastgelegd is.

9.3 ONTWERPEN SYSTEEMDEEL

Deze beroepstaak staat niet beschreven in het afstudeerplan, maar is wel uitgevoerd. Deze beroepstaak is behaald op niveau 3 door het ontwerpen van het Model binnen het systeem en de verschillende implementaties van de beschreven use cases. Deze beroepstaak is zelfstandig uitgevoerd met een lastige complexiteit.
Door gebruik te maken van Data Access Layers (DAL) om data uit een data bron op te halen en vervolgens op te slaan in een Singleton instantie van het Model, is een herbruikbare manier van veilig data ophalen waarmee men zeker is dat dezelfde data in het complete systeem gebruikt wordt.
Het implementeren van de verschillende DAL's heeft ertoe geleid dat uitbreidbaarheid van beide applicaties toegenomen is. Elke binnengekomen data wordt eerst door het DAL gehaald, waarmee de data in één formaat geladen wordt. Doordat gebruik gemaakt wordt van de verschillende DAL's en de implementatie van MVP is Loose Coupling ontstaan.

Alle communicatie vanuit de View naar de Presenter wordt asynchroon afgehandeld, waardoor de gebruikersinterface nooit vast loopt doordat veel data verwerkt moet worden. De Presenter is in staat om de View feedback te geven over de status van de verwerking zodat de gebruiker weet waar hij/zij aan toe is. De gebruikersinterface is geïmplementeerd met behulp van de stijlgids van Centric, waarbij gelet moet worden op indeling, kleuren en elementen die aanwezig dienen te zijn.

9.4 BOUWEN APPLICATIE

Volgens het afstudeerplan wordt deze beroepstaak behaald op niveau 3.
Deze beroepstaak is behaald op niveau 3 door het implementeren van het ontworpen systeem. Het systeem is gebouwd in C# en in het .NET framework versie 3.5. Om de gebruikersinterface op te bouwen is gebruik gemaakt van UserControls die door Centric ontwikkeld zijn. Verder is gebruik gemaakt van de Team Foundation Server (TFS) van Centric om het versiebeheer af te handelen.
De gebruikte ontwikkelomgeving is Visual Studio 2008.

Het systeem is modulair opgebouwd, waardoor code zo herbruikbaar mogelijk is gemaakt. Functionaliteit die in zowel de Viewer als in de Extractor applicatie gebruikt is, is in een gedeeld Dynamic-link library (dll) bestand geplaatst, zodat de code maar één keer geschreven hoeft te worden. Door het toepassen van Test Driven Development is het systeem te testen.
De databaseconnectie naar Oracle zit in een externe dll en maakt gebruik van de externe tool DotConnect. Het ophalen van data wordt gedaan via methodes van DotConnect, waardoor geen SQL-query's in het systeem zijn, zodat wanneer een nieuwe versie van Oracle uitkomt enkel een nieuwe versie van DotConnect geïnstalleerd dient te worden om de nieuwe versie van Oracle te gebruiken.

9.5 INITIËREN EN PLANNEN VAN HET TESTPROCES

In het afstudeerplan staat beschreven dat deze beroepstaak op niveau 3 behaald dient te worden tijdens het afstudeerproject. Deze beroepstaak is volgens niveau 3 tijdens het afstudeerproject behaald. Tijdens het project is een testplan opgesteld, het testplan definieert de verschillende tests die uitgevoerd moeten worden voordat een release kan worden uitgebracht.
Het testplan definieert verschillende systeemtests en acceptatietests.
Onder de systeemtests vallen schermtests en consistentietests. Onder acceptatietests vallen gebruikersacceptatietests en performancetests.
Gebruikersacceptatietests zijn gedefinieerd in de testscripts. De testscripts zijn opgesteld aan de hand van een template die gebruikt wordt door het testteam van de MVV afdeling bij Centric.

De testscripts, het testplan, de testrisicoanalyse en het testrapport zijn gecontroleerd en uitgevoerd door het testteam van de afdeling MVV bij Centric.
TMap is als basis gebruikt tijdens het initiëren van het testproces.
Het testframework tijdens het project is Visual Studio 2008, deze applicatie ondersteund unit-tests en biedt de mogelijkheid om alle tests van het complete systeem in één keer uit te voeren.

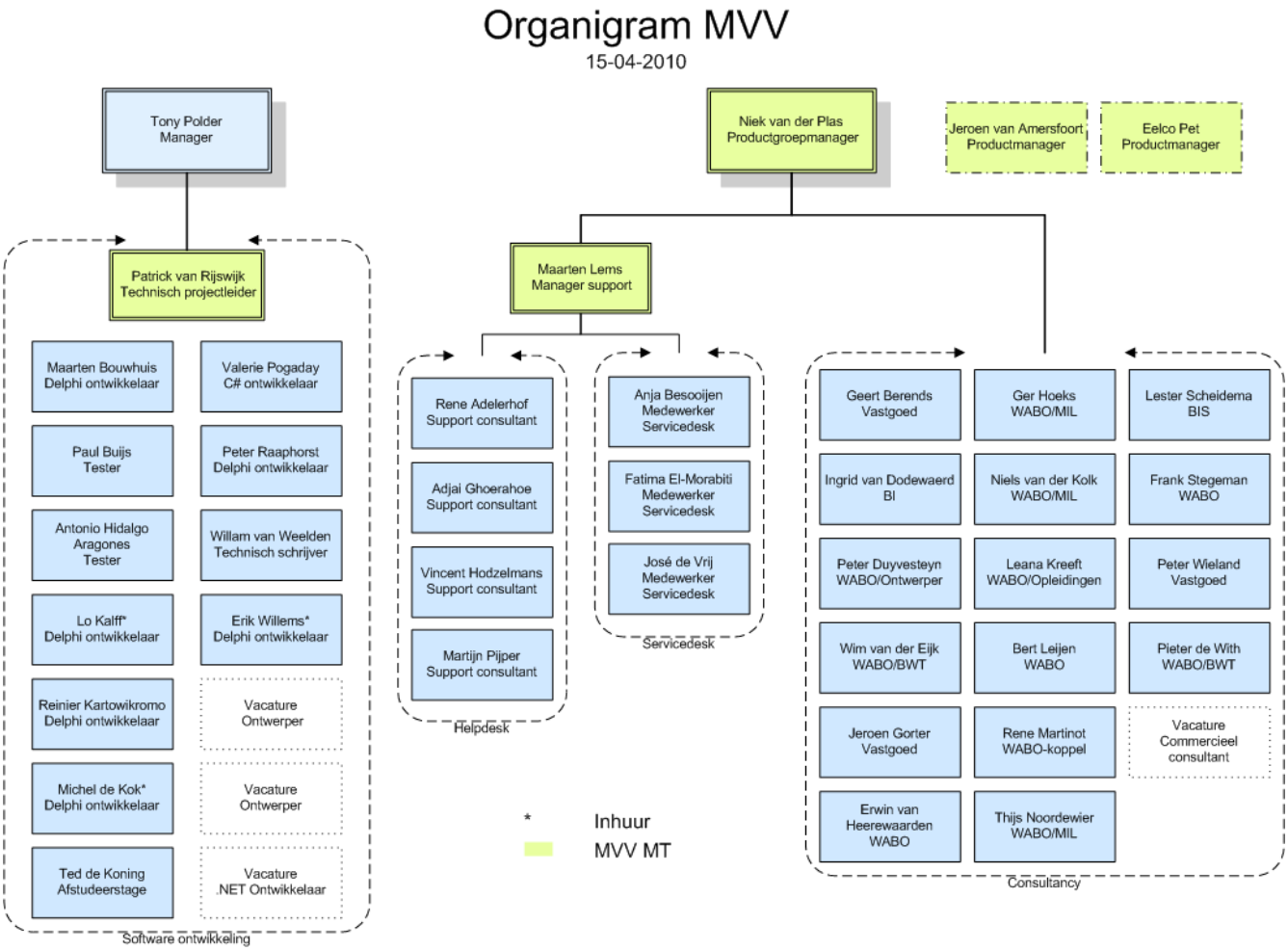
Het hele testproces is gecontroleerd door het testteam van Centric.

9.6 UITVOEREN VAN EN RAPPORTEREN OVER HET TESTPROCES

Volgens het afstudeerplan wordt de beroepstaak "Uitvoeren van en rapporteren over het testproces" op niveau 3 uitgevoerd. Deze beroepstaak heb ik op niveau 3 behaald door het opstellen van testscripts, het laten uitvoeren van de testscripts door het testteam van Centric en het analyseren van de testresultaten van de uitgevoerde tests. De beroepstaak is zelfstandig uitgevoerd.
Het testteam bij Centric heeft de tests uitgevoerd die beschreven zijn in het testplan. Tijdens het testen zijn een aantal bevindingen gevonden waarvan het prioriteitniveau bepaald is. De prioriteit is vastgesteld door te analyseren wat voor effect de bevinding op het systeem of op andere systemen heeft.
De performancetests bevatten, per performancetest, testresultaten van meerdere testruns met bijbehorende analyse van de resultaten en conclusie.
Het testrapport is goedgekeurd door het testteam bij Centric.

10. INTERNE BIJLAGEN

INTERNE BIJLAGE 1: ORGANIGRAM



INTERNE BIJLAGE 2: BEGRIPPENLIJST

Begrip	Toelichting
MVP	Staat voor Model-View-Presenter, een software-architectuur waarbij de logica verdeeld is in drie delen, het Model, de View en de Presenter.
DAL	Staat voor Data Access Layer. Een DAL zorgt voor het ophalen of wegschrijven van data van of naar een databron.
BLL	Staat voor Business Logic Layer. Het BLL zorgt voor de logica dat in een bepaalde context hoort. In het systeem heeft elke use case een BLL.
Unit-test	Geautomatiseerde white-box test om functionaliteit onafhankelijk te testen.
DBMS	Staat voor databasemanagementsysteem. Een DBMS is een systeem waar databases in beheerd kunnen worden.
DotConnect	DotConnect is een .NET library waarmee verbinding naar een Oracle DBMS gemaakt kan worden. DotConnect is ontwikkeld door DevArt.
DevExpress	DevExpress is een library waarin allerlei elementen zitten die de windows.forms elementen uitbreiden met extra functionaliteit.
Centric Framework	Applicatieframework ontwikkeld door Centric in C#.NET.
Centric Controls	Centric Controls is een library van elementen die de DevExpress elementen uitbreiden met de Centric huisstijl en extra functionaliteit toevoegen.
TMap	TMap staat voor Test Management Approach. TMap is een testaanpak ontwikkeld door het Nederlanse Sogeti.
XML	XML staat voor Extensible Markup Language. XML is een standaard om gegevens gestructureerd weer te geven of op te slaan. XML wordt opgeslagen als platte tekst.

INTERNE BIJLAGE 3: LITERATUURLIJST

BOEKEN:

- "Pro C# with .NET 3.0" door Andrew Troelsen
- "Testen volgens TMap" (2e druk) door Martin Pol, Ruud Teunissen en Erik van Veenendaal

GERAADPLEEGDE WEBSITES

- http://www.ambysoft.com/unifiedprocess/agileUP.html
- http://msdn.microsoft.com/en-us/library/cc540792.aspx
- http://blog.vuscode.com/malovicn/archive/2007/10/25/model-view-presenter-mvp-design-pattern-close-look-part-1-passive-view.aspx
- http://www.devart.com/dotconnect/oracle/docs/
- http://siliconwhisperer.com/2007/12/three-tier-web-application-framework/
- http://siliconwhisperer.com/2007/12/the-model-view-controller-mvc-architecture/

11. EXTERNE BIJLAGEN

Plan van aanpak

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

INHOUDSOPGAVE

INLEIDING

Dit document is opgesteld naar aanleiding van voor het afstudeerproject bij Centric.

Het document is bedoeld om te beschrijven hoe het project aangepakt gaat worden. In dit document wordt de opdracht beschreven, de afbakening van de opdracht en de globale planning.
Na het lezen van dit document heeft u de kennis over de aanpak van het project.

OPDRACHTSOMSCHRIJVING

BEDRIJF

Centric IT Solutions is een bedrijf dat aan veel verschillende soorten bedrijven en overheidsinstanties oplossingen door middel van software aanbiedt.
Veel van de software die Centric levert, is te koppelen met andere applicaties van Centric.
Naast het leveren van standaard software pakketten, ontwikkelt Centric ook maatwerk indien er specifieke wensen zijn van de klant.

Het ontwikkelen van applicaties en maatwerk wordt door medewerkers van Centric zelf gedaan. Centric maakt geen gebruik van outsourcing omdat de eigenaar van Centric de kennis intern wenst te houden.

De afdeling waar de afstudeeropdracht wordt uitgevoerd heeft vooral lokale overheden als klanten. De applicaties die op deze afdeling ontwikkeld worden zijn gericht op bouw- en woningtoezicht, bouwvergunningen en milieuvergunningen. Op de afdeling worden applicaties ontworpen, gebouwd en getest, tevens worden systeembeheerders begeleid om systemen in te richten of updates uit te voeren. Daarnaast worden ook trainingen gegeven aan eindgebruikers van de software.
Tevens is er op de afdeling een helpdesk aanwezig, waar eerste- en tweedelijns ondersteuning aan eindgebruikers geleverd wordt.

AANLEIDING

Vanuit Centric is er vraag naar een manier waarmee databasestructuren eenvoudig weergegeven kunnen worden. De gegenereerde overzichten zullen gebruikt worden om maatwerk te vereenvoudigen en om systeembeheerders bij klanten te ondersteunen.

PROBLEEMSTELLING

Op dit moment kunnen medewerkers van Centric alleen de structuur van databases bekijken door middel van de interface van het Oracle DBMS en is er geen manier om relaties tussen verschillende databases in te zien. Tevens wordt er regelmatig van de klant verzocht om een databasediagram van de applicaties te versturen, dit is op dit moment niet mogelijk.

DOELSTELLING VAN DE AFSTUDEEROPDRACHT

Het hebben van een systeem wat inzicht biedt in relaties van verschillende databases, die tevens met andere databases relaties kunnen hebben.
Het op te leveren systeem dient generiek voor Oracle databases toepasbaar te zijn.

SYSTEEMEISEN

Het systeem moet databasediagrammen kunnen genereren aan de hand van Oracle databases. Deze databases kunnen gekoppeld zijn aan andere databases, hier moet de applicatie rekening mee houden. De gegenereerde databasediagrammen moeten exporteerbaar zijn naar een XML bestand. De geëxporteerde databasediagrammen moeten importeerbaar zijn in de applicatie, zodat de diagrammen weer worden weergegeven.

Er zullen twee applicaties ontwikkeld worden. Één van de applicaties regelt het uitlezen van databases,dit onderdeel wordt in dit document als "Extractor" aangeduid.De Extractor krijgt de functionaliteit om een XML bestand te genereren waar de databasestructuur in opgeslagen is.
De gegenereerde XML bestanden worden geïmporteerd in de "Viewer", zodat er een databasediagram op het scherm zichtbaar wordt.

AFBAKENING

De Viewer kan enkel inzicht bieden in de databasestructuur. De databasediagrammen worden opgebouwd aan de hand van Oracle databases die door middel van database links met elkaar verbonden worden. De diagrammen moeten exporteerbaar zijn naar een XML formaat.

Het mogelijk maken om de databasediagrammen te wijzigen valt niet binnen de opdracht.
Het uitlezen van databases uit een ander database management systeem dan Oracle valt niet binnen de opdracht.

WERKZAAMHEDEN

Tijdens de inception fase worden de requirements van de applicaties vastgelegd. Deze requirements zullen de rest van het ontwerp bepalen.

Naast de requirements worden er ook use case diagrammen, met bijbehorende beschrijvingen, gemaakt. In deze use cases worden de acties die mogelijk zijn binnen de applicatie vastgelegd.

Binnen de elaboration fase wordt de nadruk op het ontwerpen van de architectuur van de applicaties gelegd. Er worden klassediagrammen gemaakt waardoor de structuur van de applicaties vastgelegd worden. Tevens worden er sequence diagrammen gemaakt, zodat de interactie tussen de verschillende componenten wordt gedefinieerd.

Er wordt er een testplan opgesteld. Het testplan bevat onder andere een testrisicoanalyse, testdoelen en de soorten tests die uitgevoerd moeten worden.

Tijdens de Construction fase zal het implementeren van het ontwerp, dat is ontstaan tijdens de inception- en elaboration fase, gerealiseerd worden. Tijdens het implementeren van het ontwerp zal gebruik worden gemaakt van Test Driven Development door middel van unit-tests.

Tevens wordt in deze fase het testplan uitgevoerd en worden er PSD’s gemaakt.

GLOBALE PLANNING

Het project duurt 17 weken. In deze 17 weken zal het volgende opgeleverd worden:

Activiteit	Aanvang	Deadline
Opstellen use cases	week 1	week 2
Opstellen requirements	week 2	week 2
Opstellen domeinmodel	week 3	week 4
Ontwerpen sequence diagram	week 3	week 4
Ontwerpen klassediagram	week 3	week 4
Ontwerpen deployment diagram	week 3	week 4
Opstellen van testplan	week 4	week 5
Bouwen van applicatie (extractor)	week 5	week 9
Maken van Unit-tests	week 5	week 14
Maken van PSD's	week 5	week 14
Bouwen van applicatie (viewer)	week 10	week 14
Uitvoeren testplan	week 12	week 13
Opstellen eindverslag	week 13	week 17

MIJLPAALPRODUCTEN

De volgende mijlpaalproducten worden tijdens het afstudeerproject aan de opdrachtgever opgeleverd.

- Plan van aanpak
- Inception rapport
    - Use cases
    - Requirements
- Elaboration rapport
    - klassediagram
    - deployment diagram
    - sequence diagram
- Testplan
- Construction rapport
    - Testrapport
    - PSD's
- Applicatie
- Unit-tests

Inception rapport

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

INHOUDSOPGAVE

INLEIDING

Dit document is geschreven naar aanleiding van het afstudeerproject bij Centric. Tijdens het afstudeerproject wordt een softwaresysteem ontwikkeld waarmee de metadata van Oracle databases als databasediagram op het scherm getoond kan worden.

Dit document beschrijft de bevindingen tijdens de Inception fase en de te realiseren functionaliteit.
Dit document bevat de requirements, use cases en het domeinmodel die opgesteld zijn aan de hand van de eisen van de opdrachtgever. Dit document wordt gebruikt om aan te tonen welke functionaliteit er geïmplementeerd gaat worden in de applicaties die gebouwd gaan worden.

1. OPDRACHT

Op het moment van schrijven kunnen medewerkers van Centric alleen de structuur van databases bekijken door middel van de interface van de Oracle DBMS. Tevens komt er regelmatig van de klant een verzoek om een databasediagram van de Oracle databases die bij de Centric applicaties horen te versturen, iets wat op dit moment niet mogelijk is.

Vanuit Centric is de opdracht gegeven om een systeem te ontwikkelen waarmee databasediagrammen gegenereerd worden aan de hand van Oracle databases.
De belangrijkste functionaliteit van het systeem is het omzetten van een databasestructuur naar een databasediagram.

Het systeem is opgedeeld in twee applicaties. De eerste applicatie is de Extractor, deze applicatie zorgt voor het uitlezen van de databasestructuur en het genereren van een XML bestand waar deze data in staat. De tweede applicatie is de Viewer, deze applicatie zorgt voor het omzetten van een XML-bestand naar een visueel databasediagram.

De Extractor is bedoeld voor intern gebruik. Applicatiebeheerders gebruiken deze applicatie om de structuur van een database uit te lezen en een XML-bestand ervan te maken.

De Viewer is bedoeld voor zowel intern als extern gebruik. Intern is het vooral bedoeld om softwareontwikkelaars te ondersteunen tijdens onderhoud aan een applicatie of database. Extern is de applicatie vooral bedoeld voor systeembeheerders bij klanten. Centric krijgt van klanten regelmatig verzoeken naar een overzicht van de databasestructuur.

2. REQUIREMENTS

Dit hoofdstuk beschrijft de requirements van het te realiseren systeem. De requirements zijn opgesteld aan de hand van informatie verkregen van de opdrachtgever en beschrijven de te implementeren functionaliteit.

2.1 FUNCTIONELE EISEN

2.1.1 Algemene eisen

- Beide applicaties moeten los van elkaar bruikbaar zijn

2.1.2 SPECIFIEKE EISEN

In dit deel van dit document worden de specifieke eisen, die aan het systeem worden gesteld, gedefinieerd.
Het systeem is opgedeeld in twee applicaties. De eerste applicatie is de Extractor, deze applicatie maakt verbinding met de gewenste database om de structuur uit te lezen.
De tweede applicatie is de Viewer, deze applicatie gebruikt, door de Extractor gegenereerde, XML-bestanden om een database diagram te genereren. De databasediagrammen die door de Viewer gegenereerd worden, moeten op het scherm getoond worden.

Extractor

- De Extractor moet de entiteiten "Tabel", "View" en "Function" met bijbehorende kolommen en referenties kunnen importeren uit Oracle databases.
- De Extractor moet een XML-bestand kunnen maken aan de hand van de structuur van Oracle databases.

Viewer

- De Viewer moet XML-bestanden kunnen importeren die gegenereerd worden door de Extractor.
- De Viewer moet databasediagrammen op het scherm kunnen tonen aan de hand van een geïmporteerd XML-bestand.
- De Viewer moet een database diagram kunnen exporteren naar XML, volgens dezelfde XML-structuur als de Extractor.
- De tabellen, views en functions in de databasediagrammen die gegenereerd zijn moeten middels een sleepactie van locatie kunnen wijzigen.

Beide applicaties moeten gebruik maken van het Centric framework.

2.1.3 USER-INTERFACE

De user-interface is het gedeelte van het systeem dat de gebruiker ziet wanneer het systeem is opgestart.

- De user-interface moet gebruik maken van componenten die door Centric ontwikkeld zijn
- De applicatie moet gebruik maken van Windows forms

TEKSTEN

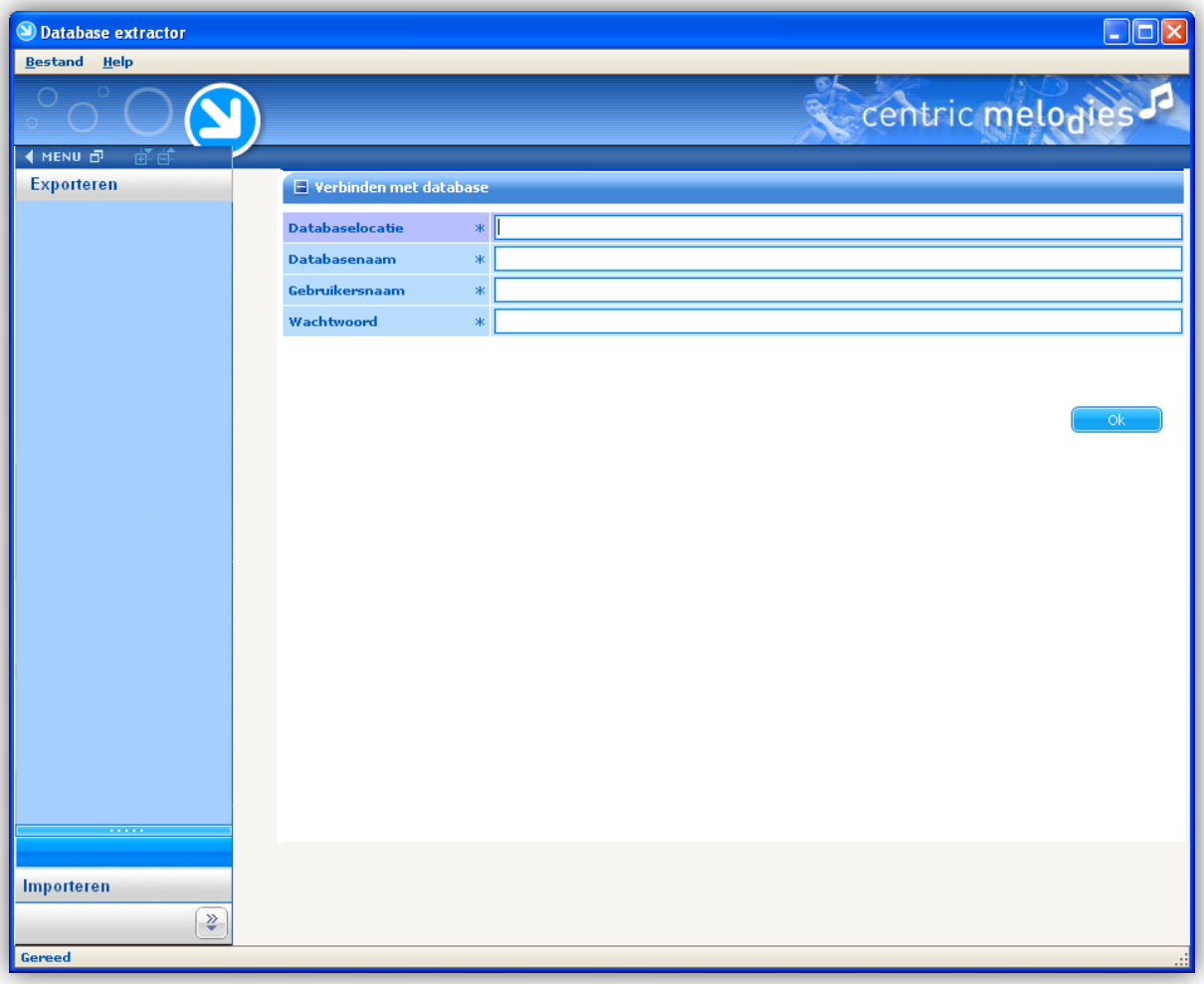
Teksten in beide applicaties worden in het Nederlands weergegeven.

MOCKUPS

Om een indruk te geven van de indeling van de user-interface zijn er een aantal mock-ups gemaakt.
Mock-ups geven een indruk in het uiterlijk van een gebruikersinterface of een ander grafisch deel van een systeem. Er worden menu-items gebruikt die overeenkomen met de uitstraling van andere Centric applicaties. De menu-items zijn door Centric ontwikkeld en worden beschikbaar gesteld voor dit project.

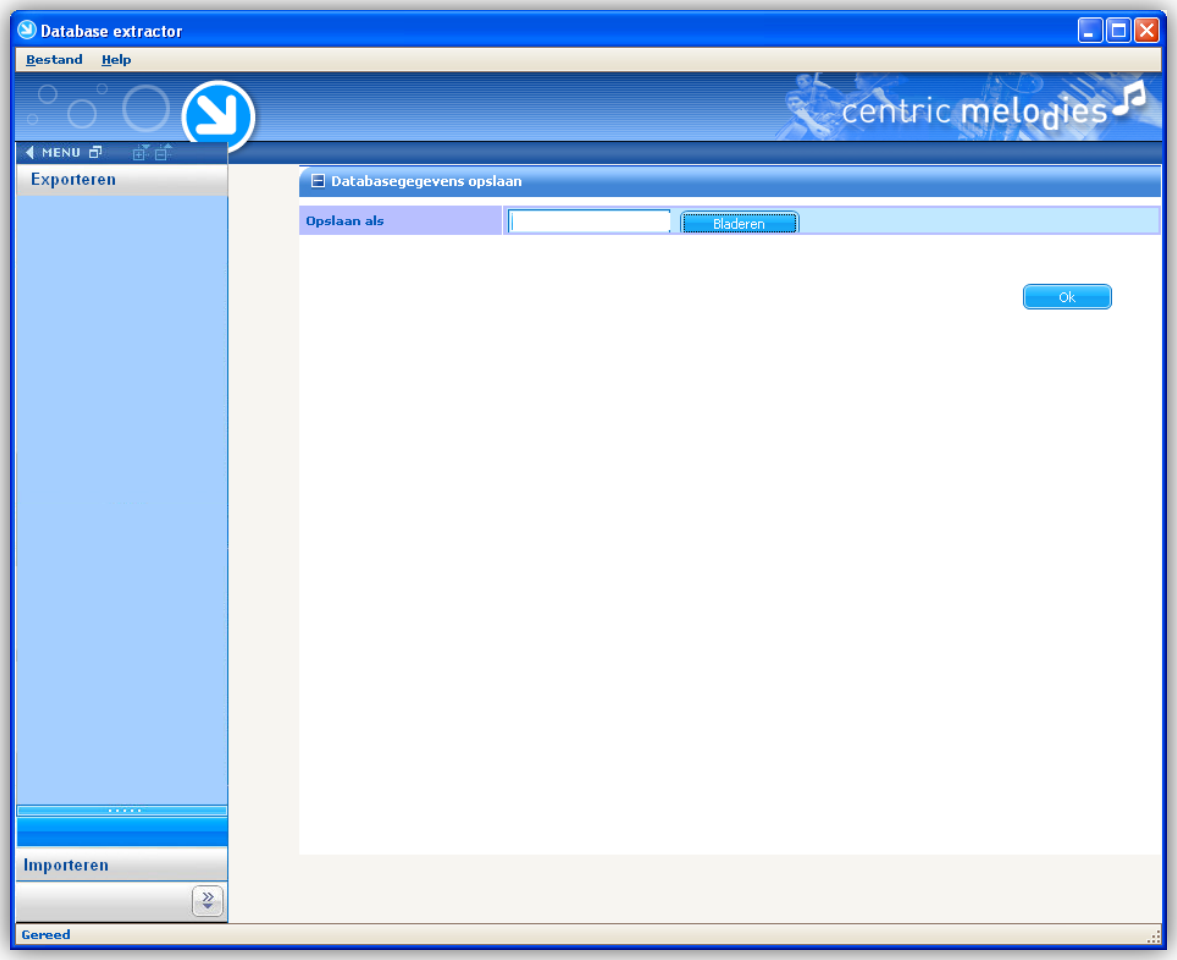
EXTRACTOR

De extractor wordt gebruikt om de structuur van een database uit te lezen, om vervolgens deze data te exporteren naar een XML-bestand.
De gebruiker moet bij het verbinden met een DBMS aangeven naar welke database verbonden moet worden.



Afbeelding 1, Mock-up extractor interface, verbinden met database.

Nadat de applicatie data uit de opgegeven database heeft ingelezen, wordt de gebruiker geconfronteerd met het volgende scherm. Het volgende scherm biedt de optie om de geïmporteerde data te exporteren naar een XML-bestand. De gebruiker kan in dit scherm een XML-bestand selecteren om vervolgens de data in op te slaan.

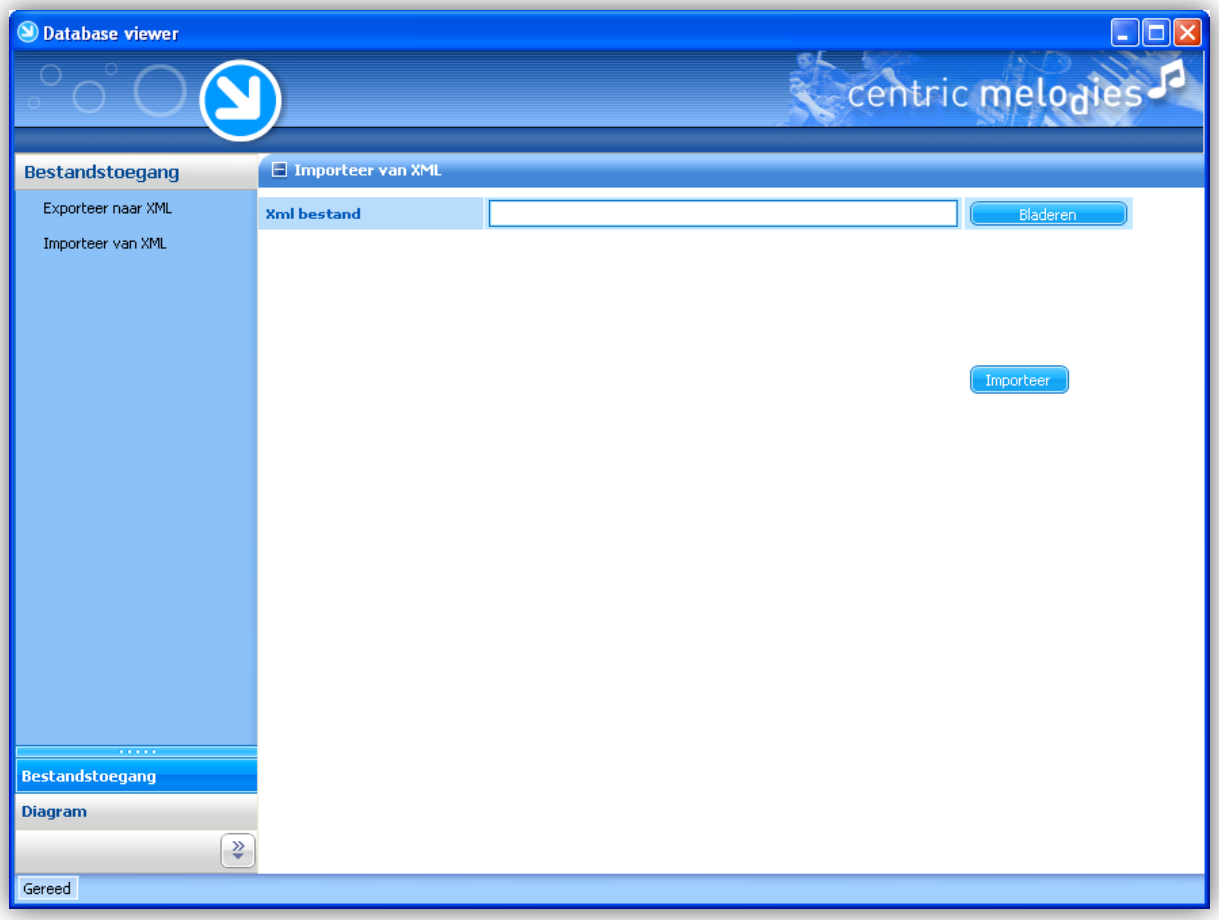


Afbeelding 2, Mock-up extractor interface, exporteren van data naar XML.

VIEWER

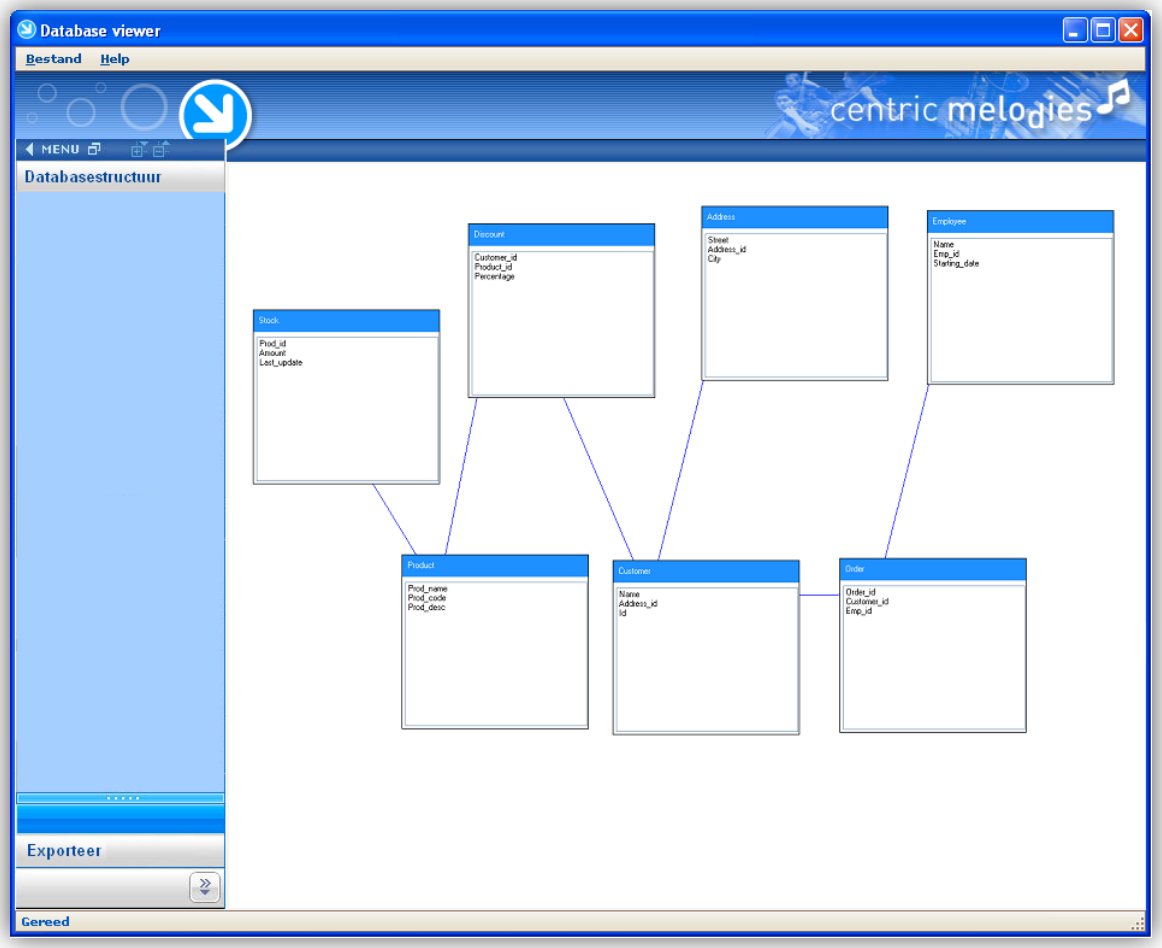
De viewer wordt gebruikt om XML-bestanden in te lezen en op basis van deze bestanden databasediagrammen te tonen op het scherm.
Het moet mogelijk zijn om XML-bestanden te importeren, maar ook te exporteren. Het exporteren is noodzakelijk voor toekomstig onderhoud aan de applicatie. Het scherm om data te exporteren is identiek aan het XML-exporteerscherm van de Extractor.

Op het moment dat de gebruiker de Viewer opstart, wordt het XML-importeerscherm getoond. Het XML-importeerscherm biedt de mogelijkheid om een XML-bestand te selecteren om vervolgens de data uit het XML-bestand in het geheugen van de Viewer te laden.



Afbeelding 3, Mock-up viewer interface

Nadat een XML-bestand succesvol is geïmporteerd, wordt de ingeladen data in diagramvorm getoond. Het volgende scherm geeft een impressie van een ingeladen diagram in de Viewer.



Afbeelding 4, Mock-up viewer interface.

2.2 NIET FUNCTIONELE EISEN

- De applicaties moeten draaien op een Windows PC.
- De gebruiker hoeft geen kennis te hebben van databasediagrammen om gebruik te kunnen maken van het systeem.
- De gebruiker hoeft geen kennis te hebben van programmeertalen om gebruik te kunnen maken van het systeem.

- Het systeem moet bestaan uit twee applicaties
- Beide applicaties moeten desktop applicaties zijn, dit betekend dat ze lokaal geïnstalleerd moeten worden.
- De gegenereerde diagrammen moeten exporteerbaar zijn

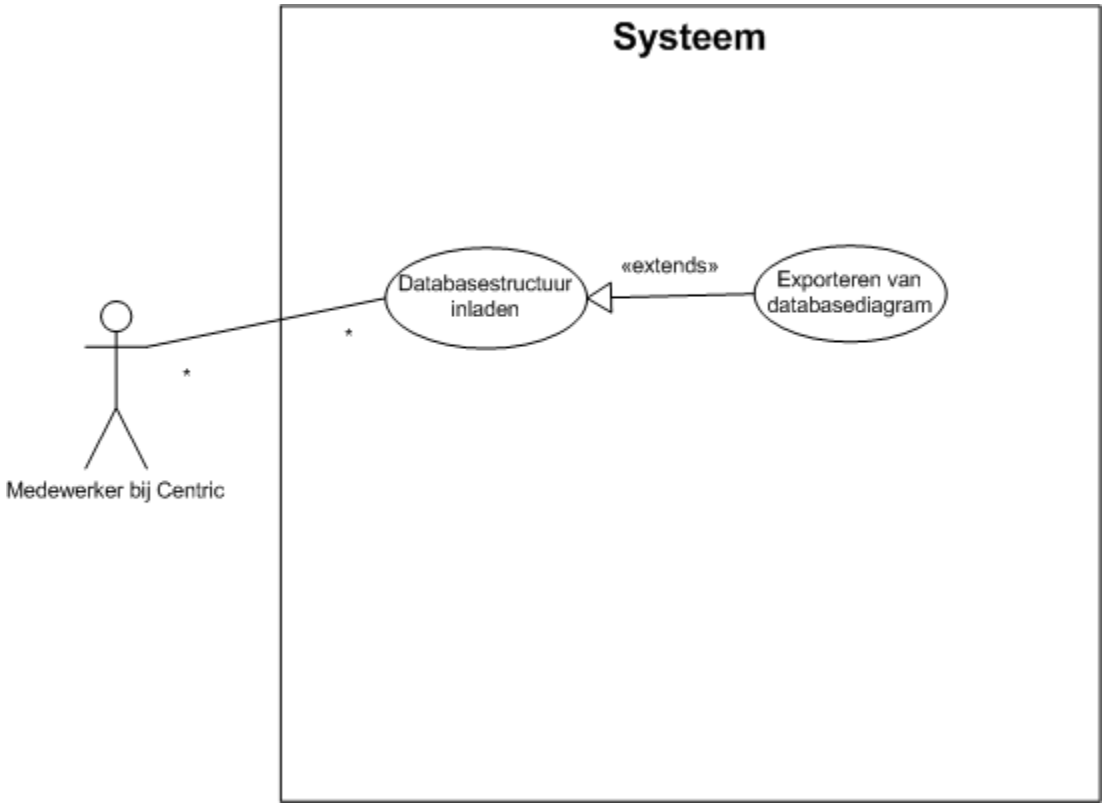
- De applicaties moeten object-georiënteerd ontwikkeld worden
- Unit-tests moeten aanwezig zijn voor de ontwikkelde classes
- Beide applicaties moeten in C#.NET ontwikkeld worden

3. USE CASES

3.1 EXTRACTOR

De Extractor heeft als primair doel data om te zetten vanuit een Oracle DBMS naar een XML-bestand. Dit hoofdstuk beschrijft de verschillende use cases binnen de Extractor.

3.1.1 USE CASE DIAGRAM



Afbeelding 5, use case diagram van de Extractor

3.1.2 USE CASE BESCHRIJVINGEN

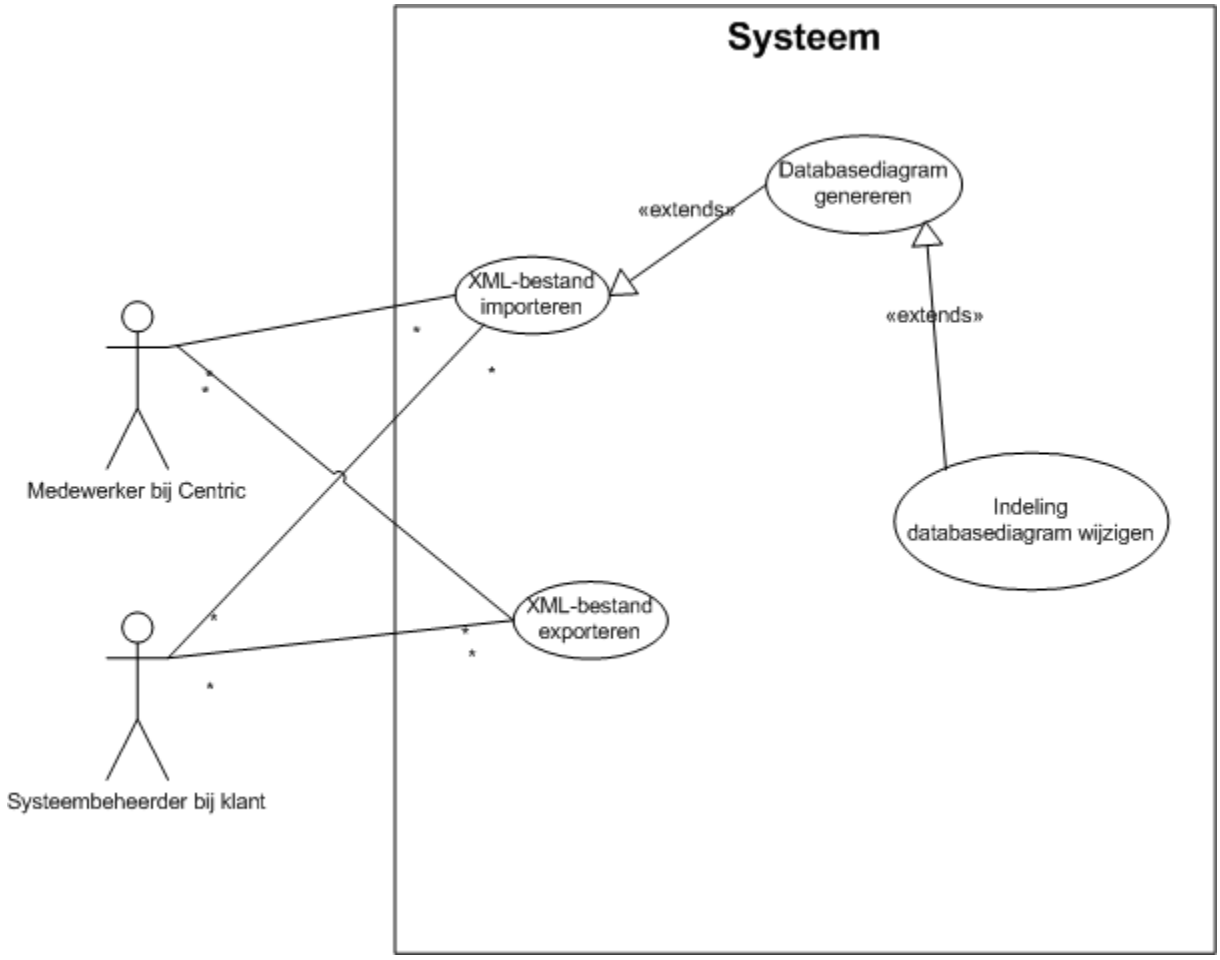
Naam	Databasestructuur inladen
Samenvatting	De structuur van een Oracle database wordt ingeladen in de applicatie.
Actoren	Medewerker bij Centric
Aannamen	De applicatie is gestart, er is nog geen verbinding met een database gemaakt.
Beschrijving	1. De actor klikt op "Verbind met database". 2. De applicatie toont een scherm waar de actor de databasenaam, server, gebruikersnaam en wachtwoord in moet vullen. 3. De actor vult de gegevens in en klikt op "Ok". 4. De applicatie laadt de structuur van de database in het geheugen. 5. Extends: Database diagram exporteren.
Uitzonderingen	• DotConnect is niet aanwezig • Gebruikersnaam en wachtwoord zijn incorrect • Geen verbinding met opgegeven database mogelijk
Resultaat	De databaseconnectie is open gezet en het exporteerscherm wordt weergegeven.

Naam	Database diagram exporteren
Samenvatting	De actor exporteert een gegenereerd XML bestand waarin de structuur staat van de opgegeven database.
Actoren	Medewerker bij Centric
Aannamen	De applicatie draait en de structuur van een database is ingeladen in het geheugen.
Beschrijving	1. De actor klikt op "Sla structuur op". 2. De applicatie toont een scherm waarin een locatie en naam voor het nieuwe bestand aangegeven kan worden. 3. De actor vult de bestandsnaam in en navigeert naar de juiste directory. 4. De actor klikt op "Ok". 5. De applicatie genereert een XML bestand van de ingeladen databasestructuur en slaat dit op in het opgegeven bestand.
Uitzonderingen	• De actor heeft geen schrijfrechten op het opgegeven bestand. • Er is geen bestand opgegeven • Het opgegeven bestand/locatie bestaat niet • Er is geen databasestructuur ingeladen
Resultaat	Er is een XML bestand gegenereerd waarin de structuur van de database opgeslagen is.

3.3 VIEWER

De Viewer heeft als primaire doel data te tonen vanuit een, door de Extractor gegenereerd, XML-bestand. De Viewer biedt de gebruiker de mogelijkheid om het gegenereerde diagram in te kunnen delen door middel van het slepen van de verschillende elementen binnen het diagram. Voor toekomstig onderhoud is het mogelijk gemaakt om XML-bestanden ook te kunnen exporteren.

3.3.1 USE CASE DIAGRAM



Afbeelding 6, use case diagram van de Viewer

3.3.2 USE CASE BESCHRIJVINGEN

Naam	XML bestand importeren
Samenvatting	De actor importeert een XML bestand waar de structuur van een database in opgeslagen is.
Actoren	Medewerker bij Centric
Aannamen	De applicatie is opgestart.
Beschrijving	1. De actor klikt op "Importeren". 2. De applicatie toont een scherm waar het juiste bestand uit het filesystem geselecteerd kan worden. 3. De actor selecteert het bestand. 4. De applicatie laadt het bestand in het geheugen. 5. De actor klikt op "Ok".
Uitzonderingen	• Het bestand is niet in het juiste formaat. • Het bestand is niet aanwezig op het filesystem
Resultaat	De data uit het XML bestand is ingeladen in de applicatie.

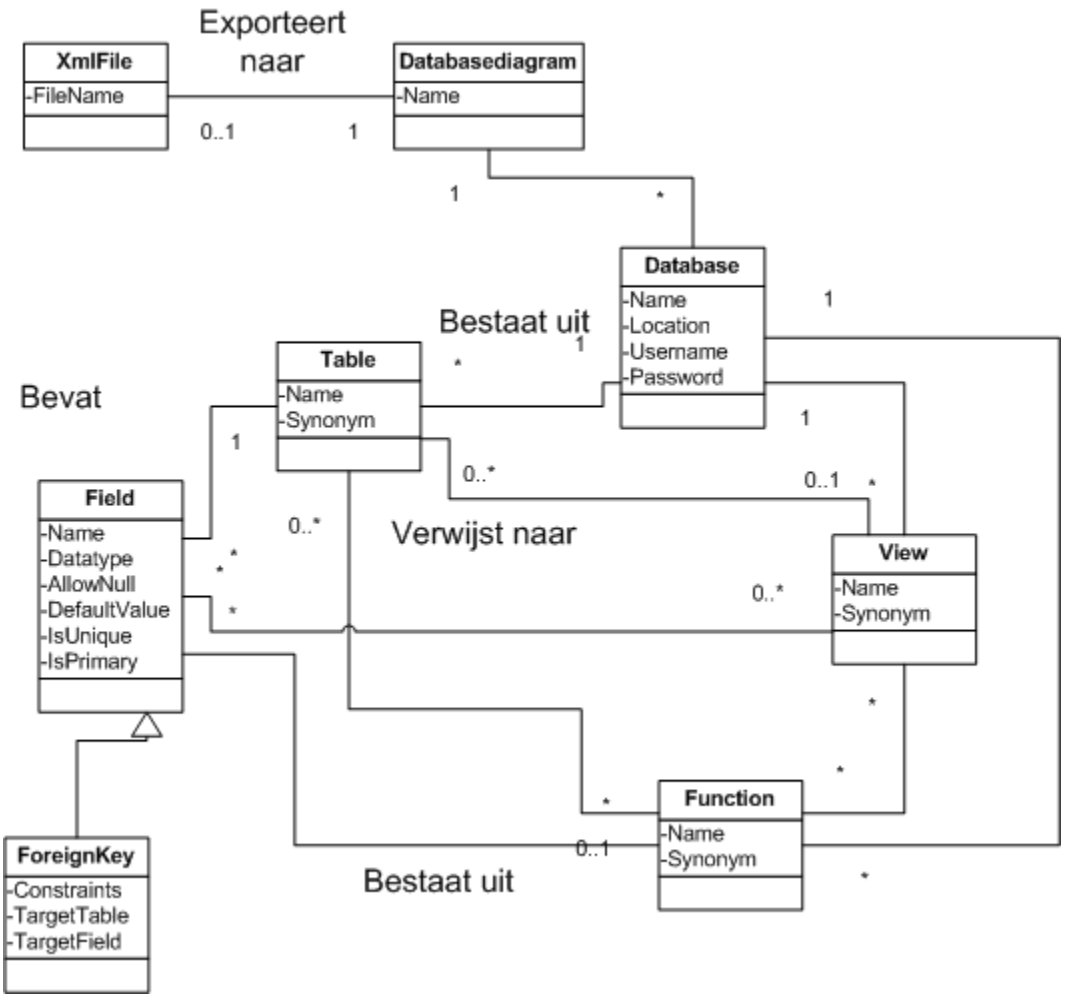
Naam	Database diagram genereren
Samenvatting	De actor wil de data van het eerder geïmporteerde XML bestand bekijken.
Actoren	Medewerker bij Centric
Aannamen	De applicatie draait en er is een XML bestand geïmporteerd.
Beschrijving	1. De actor klikt op "genereer diagram". 2. De applicatie genereert het databasediagram en toont het op het scherm.
Uitzonderingen	• Er zijn geen gegevens ingeladen in het geheugen
Resultaat	Er wordt een databasediagram weergegeven op het scherm aan de hand van de data uit het XML bestand.

Naam	XML bestand exporteren
Samenvatting	De actor wil de structuur van het eerder geïmporteerde XML bestand exporteren.
Actoren	Medewerker bij Centric
Aannamen	De applicatie draait en er is een XML bestand geïmporteerd.
Beschrijving	1. De actor klikt op "Exporteer naar XML". 2. De applicatie toont een scherm waarin een bestandslocatie en bestandsnaam opgegeven kan worden. 3. De actor vult de naam van het bestand in en navigeert naar de locatie. 4. De actor klikt op "Ok". 5. De applicatie genereert een XML bestand en schrijft deze naar de juiste locatie.
Uitzonderingen	• De applicatie heeft geen schrijfrechten op de opgegeven locatie. • Het opgegeven bestand/locatie bestaat niet • Er is geen databasestructuur ingeladen
Resultaat	Er is een XML bestand, waarin een database structuur staat, geëxporteerd naar de opgegeven locatie.

Naam	Indeling databasediagram wijzigen
Samenvatting	De actor wenst de indeling van het geïmporteerde databasediagram wijzigen
Actoren	Medewerker bij Centric
Aannamen	De applicatie draait, er is een XML bestand geïmporteerd en er is een databasediagram gegenereerd.
Beschrijving	1. De actor sleept een tabel naar een andere locatie. 2. De applicatie berekent de nieuwe coördinaten. 3. De applicatie toont de nieuwe indeling van het databasediagram.
Uitzonderingen	Geen
Resultaat	De locatie van een tabel in het databasediagram is gewijzigd en de bijbehorende relatielijnen zijn aangepast aan de nieuwe locatie.

4. DOMEINMODEL

Afbeelding 7 bevat de classes die gebruikt worden in beide applicaties. In de diagrammen, die gegenereerd worden, zijn tabellen, views en functions weergegeven. Het domeinmodel is tot stand gekomen doordat gekeken is naar de elementen die door het systeem geïmporteerd moeten worden en de bijbehorende opslag op het bestandssysteem.



Afbeelding 7, Domeinmodel van het systeem.

5. ITERATIE: OVERZICHTELIJKHEID

De "overzichtelijkheid" iteratie van het systeem implementeert functionaliteit waardoor de gebruiker een beter overzicht kan creëren van een database diagram.

5.1 OPDRACHT

De "overzichtelijkheid" iteratie omvat een grote verbetering richting een volwaardig databasediagram systeem. De opdrachtgever wenst een overzichtelijkere manier om database diagrammen weer te geven en in te delen in de Viewer applicatie.

Op het moment van schrijven kan de Viewer applicatie enkel een volledig database diagram op het scherm tonen. Wanneer het database diagram meer dan tien elementen bevat, kan het voor de gebruiker al snel onoverzichtelijk worden. De opdrachtgever wenst voor bovenstaand probleem een oplossing waardoor gebruikers een overzichtelijker diagram samen kunnen stellen.

5.2 REQUIREMENTS

5.2.1 FUNCTIONELE EISEN

EXTRACTOR

De Extractor applicatie blijft ongewijzigd in deze iteratie.

VIEWER

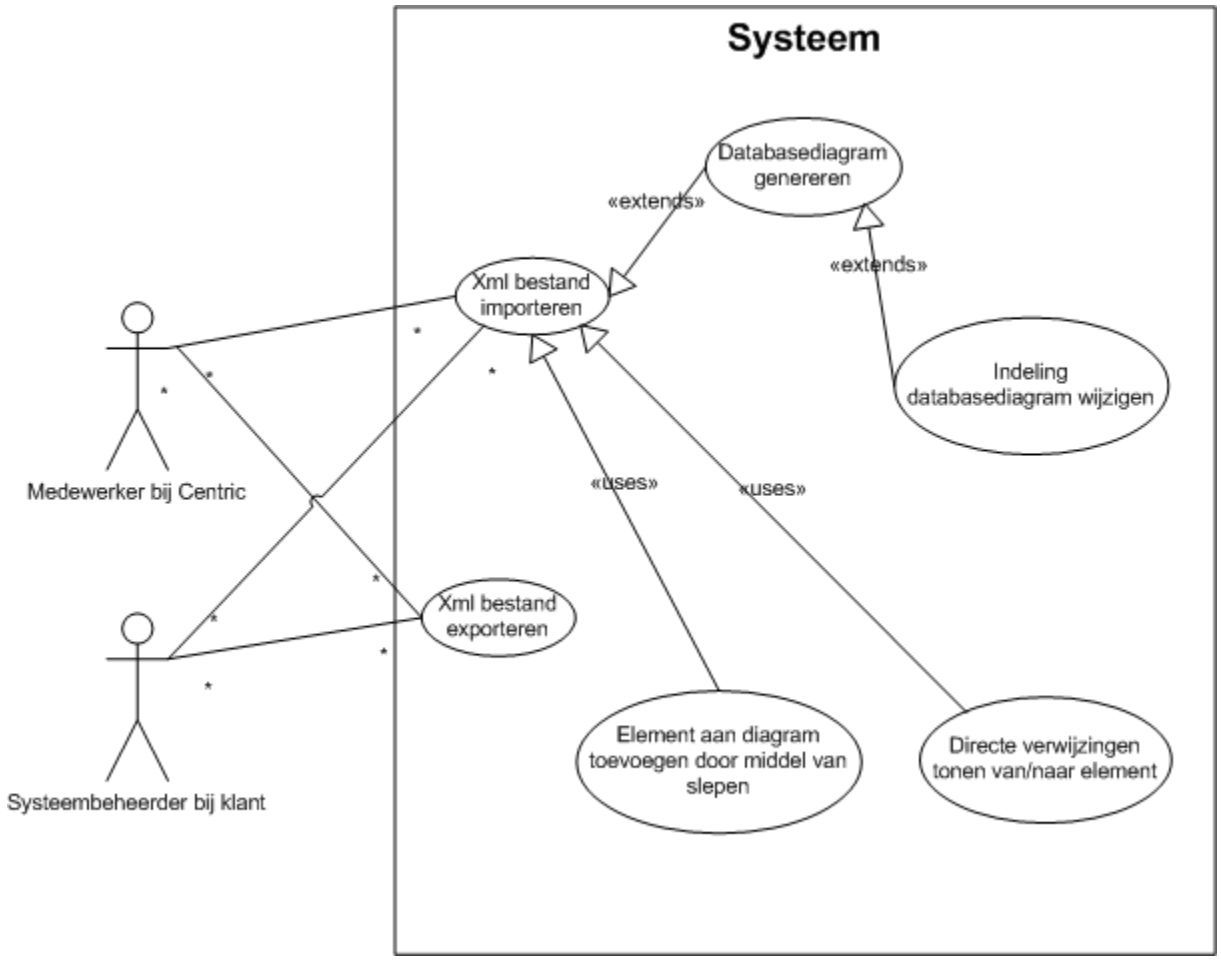
De Viewer applicatie krijgt extra functionaliteit waarmee productiviteit binnen de applicatie bevorderd wordt en waarmee de gebruiker een overzichtelijker diagram samen kan stellen.

De Viewer moet een gebruikersinterface implementeren waarmee de gebruiker elementen uit een lijst selecteert. Wanneer de gebruiker op een element binnen de lijst klikt, toont de Viewer het element waar de gebruiker op geklikt heeft en de elementen waar het geklikte element een directe verwijzing van en/of naar heeft.
Tevens moet het mogelijk zijn om elementen uit een lijst te slepen naar het database diagram veld, zodat deze individuele elementen getoond worden. Wanneer een getoond element een verwijzing heeft van en/of naar een ander getoond element, wordt deze verwijzing weergegeven.

5.2.2 NIET-FUNCTIONELE EISEN

De iteratie moet volgens dezelfde architectuur (Model View Presenter) ontwikkeld worden als de huidige geïmplementeerde functionaliteit binnen het systeem.

5.3 USE CASES



Afbeelding 8, vernieuwd use case diagram van de Viewer.

Naam	Element aan diagram toevoegen door middel van slepen.
Samenvatting	De actor wenst een element van de ingeladen database in diagramvorm weer te geven.
Actoren	Medewerker bij Centric
Aannamen	De applicatie draait, er is een XML bestand geïmporteerd en het database diagram scherm is geopend.
Beschrijving	1. De actor selecteert "Tabellen" vanuit de selectbox aan de rechterzijde van het scherm. 2. De applicatie toont alle beschikbare tabellen in een lijst. 3. De actor sleept een element uit de lijst op het diagramscherm. 4. De applicatie voegt het element toe aan het diagramscherm en toont, indien van toepassing, de verwijzingen naar andere elementen die weergegeven worden.
Uitzonderingen	• Er zijn geen tabellen beschikbaar in het ingeladen database diagram.
Resultaat	Het gesleepte element is toegevoegd aan het diagramscherm.

Naam	Directe verwijzingen tonen van/naar element.
Samenvatting	De actor wenst de directe verwijzingen van/naar een element in diagramvorm te zien.
Actoren	Medewerker bij Centric
Aannamen	De applicatie draait, er is een XML bestand geïmporteerd en het database diagram scherm is geopend.
Beschrijving	1. De actor selecteert "Tabellen" vanuit de selectbox aan de rechterzijde van het scherm. 2. De applicatie toont alle beschikbare tabellen in een lijst. 3. De actor selecteert door middel van klikken een element in de lijst. 4. De applicatie voegt het element toe aan het diagramscherm en toont, indien van toepassing, de elementen en verwijzingen die behoren bij het geselecteerde element.
Uitzonderingen	• Er zijn geen tabellen beschikbaar in het ingeladen database diagram.
Resultaat	Het geselecteerde element en de elementen en verwijzingen die behoren bij het geselecteerde element zijn toegevoegd aan het diagramscherm.

Elaboration rapport

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

INHOUDSOPGAVE

INLEIDING

Dit document is geschreven naar aanleiding van het afstudeerproject bij Centric. Tijdens het afstudeerproject wordt een softwaresysteem ontwikkeld waarmee de metadata van Oracle databases als databasediagram op het scherm getoond kan worden.

Dit document beschrijft de bevindingen tijdens de Elaboration fase.
Het hoofdstuk Architectuur bevat informatie over de architectuur van beide applicaties.
Klassendiagrammen bevinden zich in het hoofdstuk Klassendiagrammen, het is verstandig om eerst het hoofdstuk Architectuur te lezen voor meer achtergrondinformatie over de klassendiagrammen.
Tevens bevat dit document sequentiediagrammen die in de construction fase geïmplementeerd worden.

1. ARCHITECTUUR

1.1 CENTRICFX FRAMEWORK

Één van de wensen van de opdrachtgever is dat de applicatie de Centric huisstijl aanhoud. Centric gebruikt het, door Centric ontwikkelde, CentricFX framework om te zorgen dat applicaties dezelfde huisstijl aanhouden en om ontwikkelaars een hoeveelheid werk uit handen te nemen.

Het CentricFX framework omvat Centric Controls, dit zijn User controls die gebaseerd zijn op het devexpress framework en bevatten de Centric huisstijl.

1.2 MODEL VIEW PRESENTER

Er is gekozen om het design pattern Model View Presenter (MVP) te gebruiken. MVP is een variant op het Model View Controller design pattern, welke de connectie tussen de View en de Model weghaalt. Bij MVP is de View de user interface, hetgeen dat de gebruiker te zien krijgt. De Presenter handelt acties vanuit de user interface af en bevat logica als data verificatie en statemanagement. De Model vertegenwoordigd het gedeelte dat de data bevat.

Mijn implementatie van het MVP design pattern bevat een interface class genaamd IView (Zie klassendiagrammen), deze class zorgt ervoor dat er meerdere user interfaces met dezelfde onderliggende logica kunnen werken.

Het voordeel van MVP is dat de totale logica met Unit-tests te testen valt. Tevens is het mogelijk om verschillende user interfaces te maken die werken met dezelfde logica, zonder deze logica opnieuw te moeten schrijven.

1.2.1 BUSINESS LOGIC LAYER

Naast het MVP design pattern, wordt er gebruik gemaakt van een Business Logic Layer (BLL). De BLL bevat alle logica voor het verwerken van de gegeven data vanuit de Presenter. Deze extra laag is toegevoegd om de Presenter eenvoudiger te houden. Door de logica te verdelen in verschillende lagen worden de verschillende classes overzichtelijker, zodat toekomstig onderhoud eenvoudiger is.

1.2.2 DATA ACCESS LAYER

Het ophalen en wegschrijven van een databasestructuur naar XML bestanden wordt in een aparte laag geregeld, deze laag wordt het Data Access Layer genoemd (DAL).

De XmlImport en XmlExport worden zo gebouwd dat deze classes in beide applicaties bruikbaar zijn. Het is de bedoeling dat de XML import enkel een bestandsnaam nodig heeft om een Databasediagram object terug te geven waar alle data uit het opgegeven bestand in staat.

De XmlExport heeft enkel een bestandsnaam en het Databasediagram object nodig om een xml bestand aan te maken en te vullen met de data uit het Databasediagram object.

De import en export functionaliteit wordt enkel in de XmlImport en XmlExport classes verwerkt. Door deze manier van werken is het erg eenvoudig om op een nette manier andere import en export functionaliteit toe te voegen, zoals de mogelijkheid om een SQL export toe te voegen.

Het importeren van metadata uit databases is eigenlijk ook eenzelfde proces als de XmlImport, waardoor dit proces ook binnen het DAL valt. Bijkomend voordeel van het onafhankelijk maken van de classes binnen het Model is dat Loose coupling ontstaat waardoor er zonder veel moeite nieuwe importeer of exporteer functionaliteit toegevoegd kan worden.

1.2.3 HERGEBRUIK

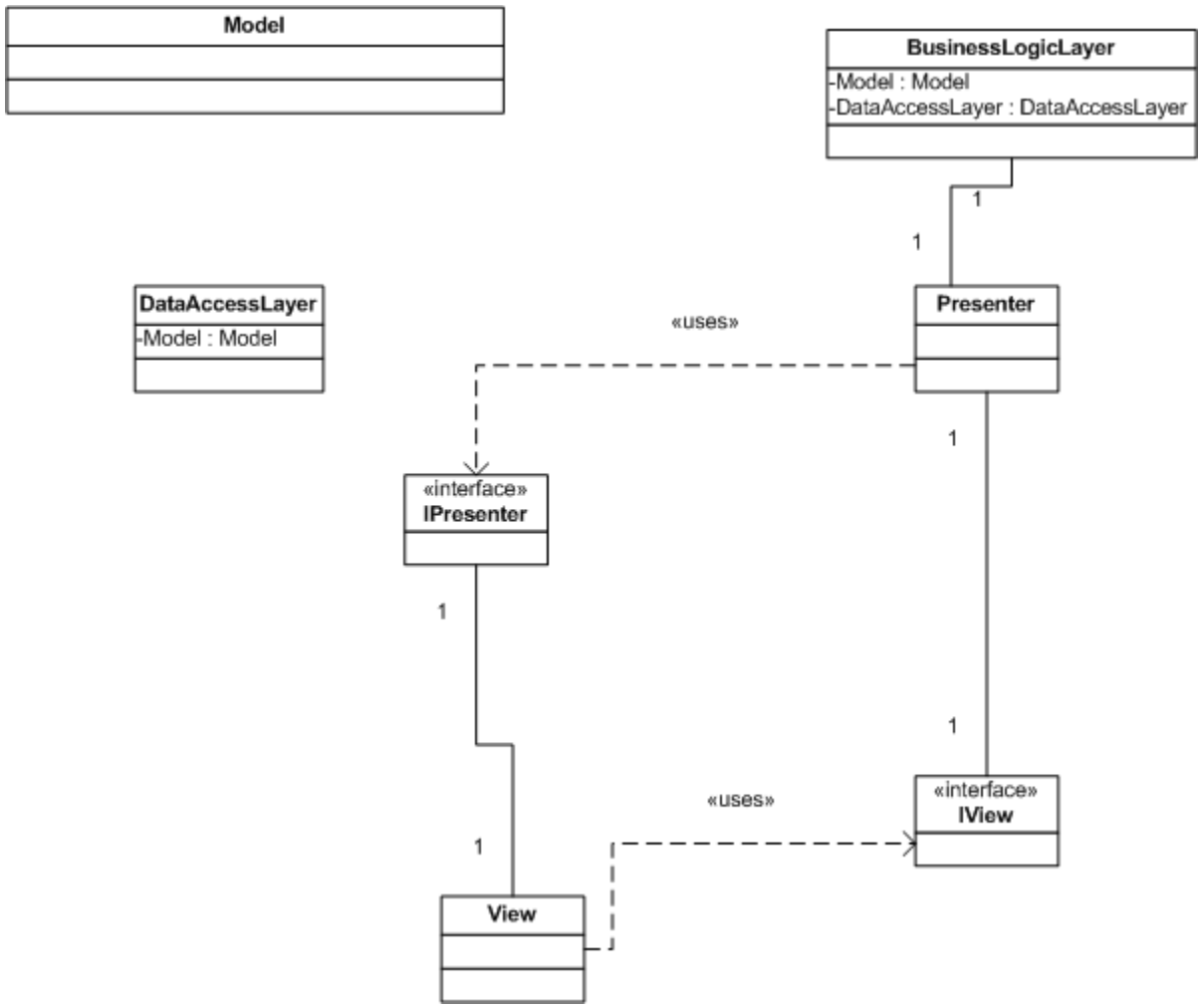
MODEL

Het Model omvat de classes die benodigd zijn om de data in het geheugen te laden. Het gaat daarbij om classes die onder de Databasediagram class vallen, voor een overzichtelijkere indruk, zie klassendiagrammen in dit document.

Om code duplicatie te voorkomen worden de classes in het Model gebruikt in beide applicaties. Het is mogelijk om deze classes op deze manier te gebruiken omdat deze zelf geen import of export specifieke functionaliteit of ingewikkelde logica bevatten. De classes binnen het Model bevatten enkel functionaliteit om data in te voegen of op te halen uit hun attributen.

1.2.4 ARCHITECTUUR IN DIAGRAM

Afbeelding 1 toont de globale structuur van beide applicaties. Het Data Access Layer (DAL) staat volledig los van andere elementen binnen de applicaties, maar bevat de logica om het Model aan te passen.

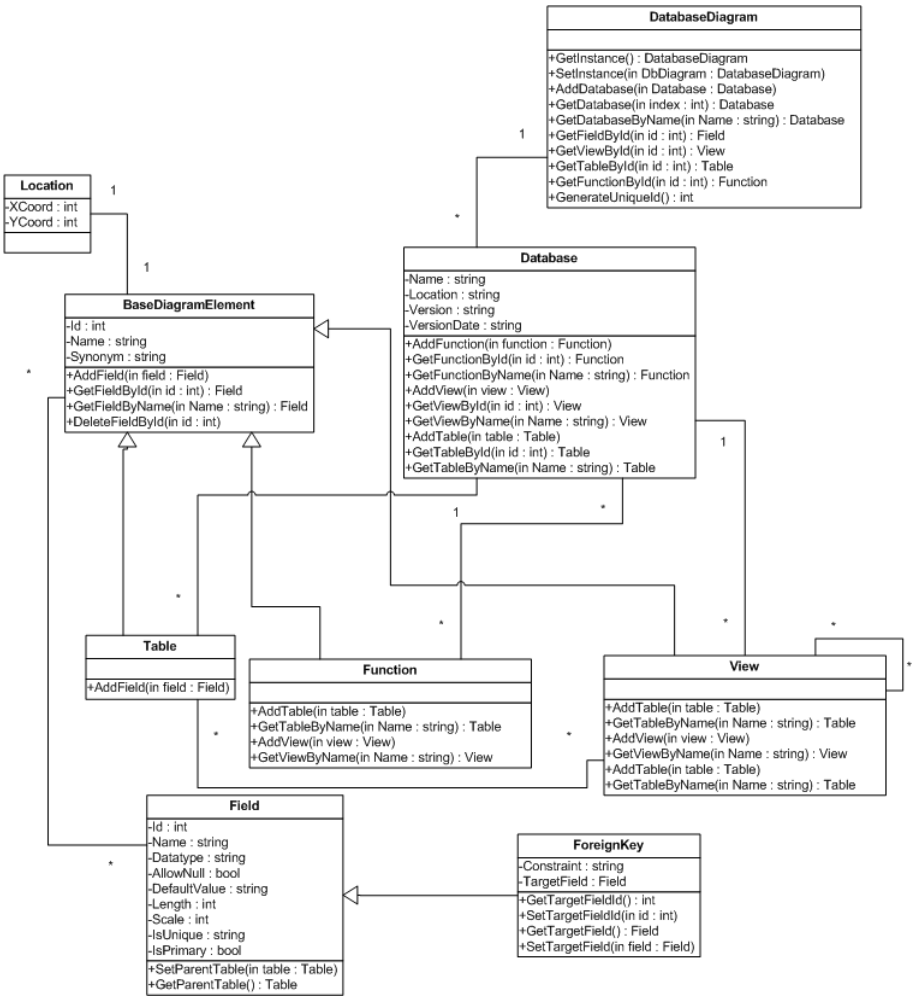


Afbeelding 1, Globale architectuur van het systeem.

2. KLASSENDIAGRAMMEN

2.1 MODEL

Het Model bestaat uit verschillende classes welke deel uitmaken van de databasestructuur. De DatabaseDiagram class is een Singleton class. De keuze om het DatabaseDiagram Singleton te maken is gemaakt doordat de data in de verschillende Presenters hetzelfde moet zijn en er nooit meer dan één DatabaseDiagram ingeladen moet worden.



Afbeelding 2, Klassendiagram van het Model.

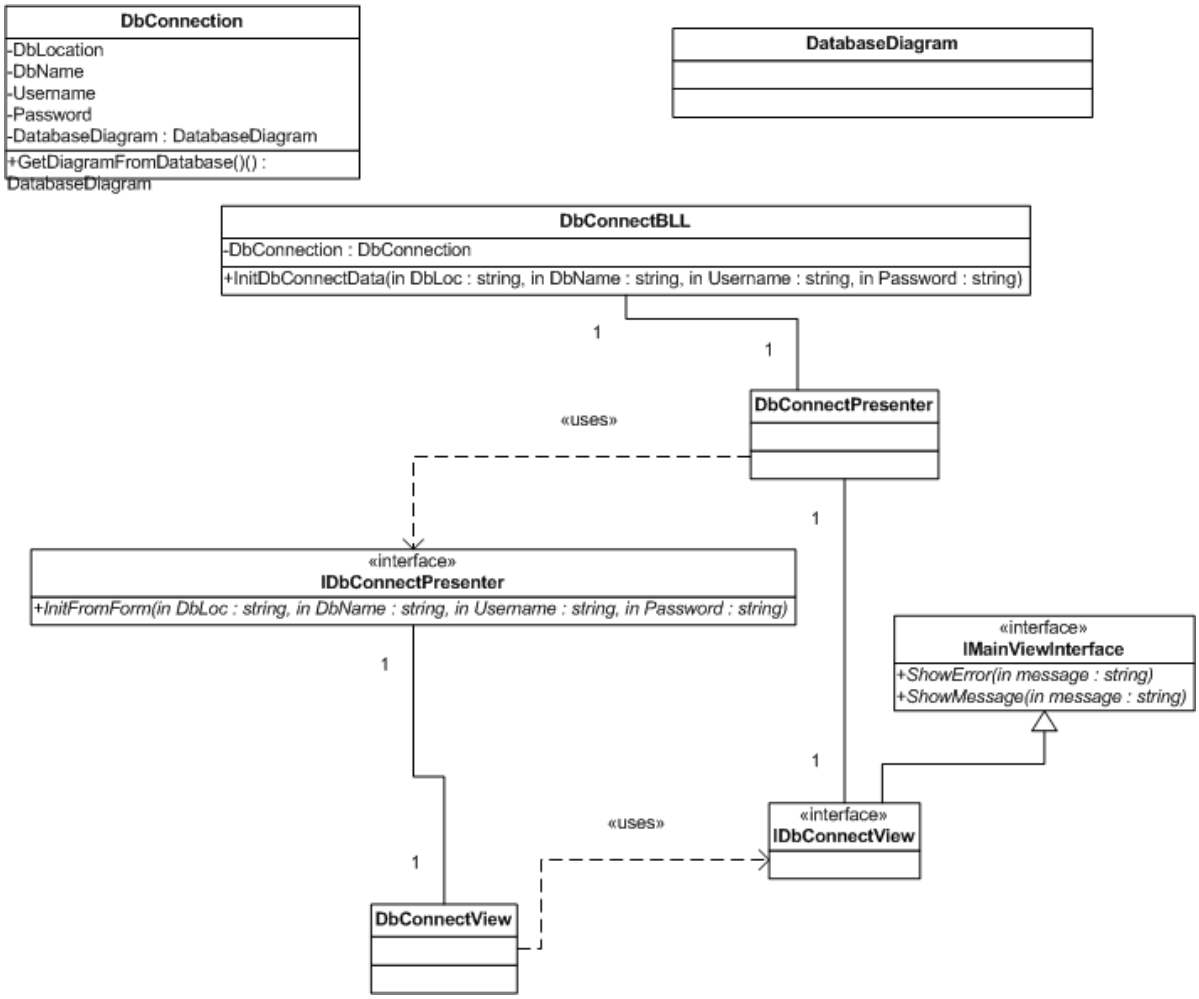
BaseDiagramElement

Er is gekozen om de classes View, Function en Table te laten overerven van de BaseDiagramElement class. Hierdoor wordt het toevoegen van nieuwe elementen en het onderhouden van de huidige elementen eenvoudiger.

2.2 EXTRACTOR

2.2.1 VERBINDEN MET DATABASE

In afbeelding 3 worden de classes getoond die te maken hebben met het ophalen van data uit een Oracle database.



Afbeelding 3, Klassendiagram van functionaliteit waarbij data vanuit een Oracle DBMS opgehaald wordt.

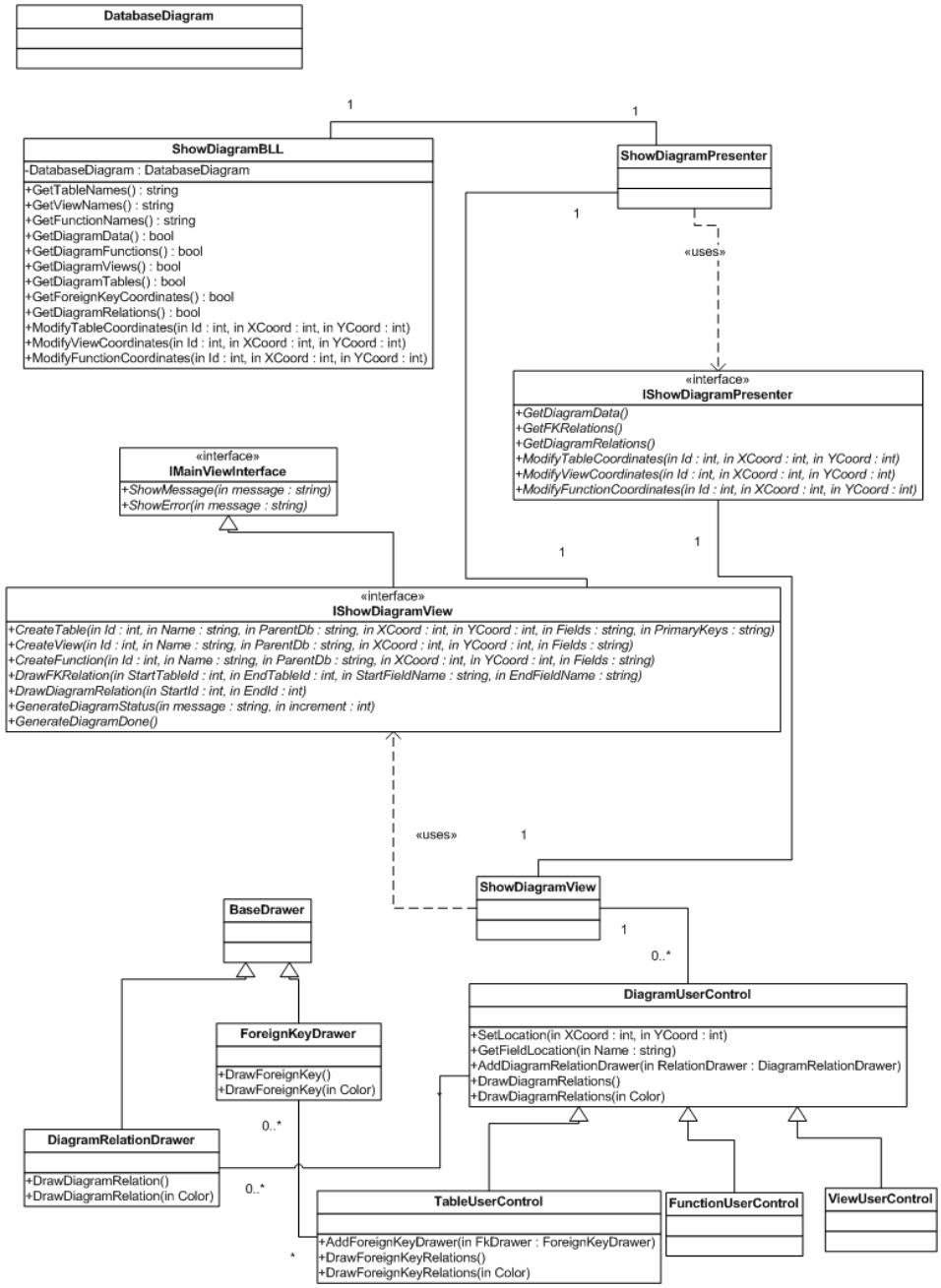
EENVOUD

Zoals afbeelding 3 al laat zien, is de architectuur van de Extractor simplistisch geïmplementeerd. Doordat er één enkele methode aangeroepen hoeft te worden, is de Extractor applicatie eenvoudig te onderhouden. Wanneer er een aanpassing aan de huidige Oracle connectie gedaan moet worden, hoeft dit in slechts één class doorgevoerd te worden.

2.3 VIEWER

2.3.1 DIAGRAM TONEN

Afbeelding 4 toont de classes die nodig zijn om een volledig databasediagram te tonen op het scherm.



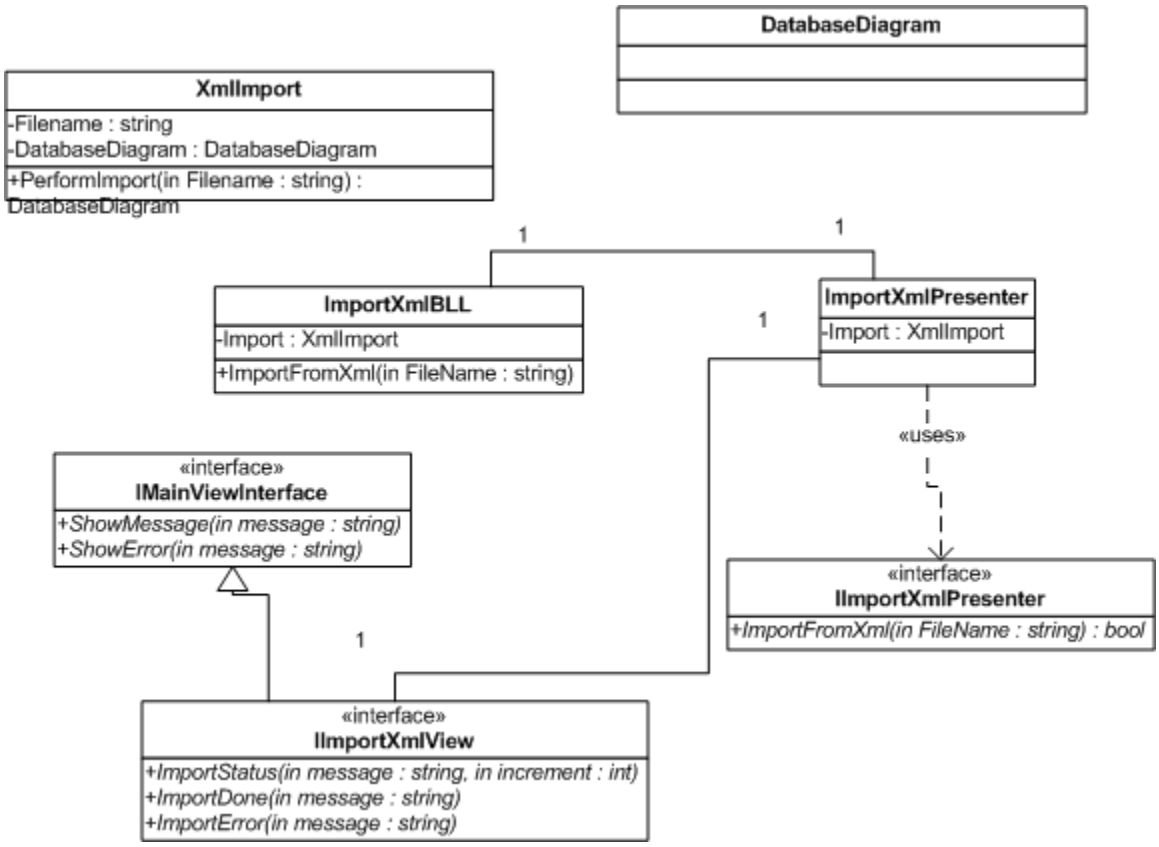
Afbeelding 4, Klassendiagram van de functionaliteit waarbij complete databasediagrammen getoond kunnen worden.

2.4 GEDEELDE FUNCTIONALITEIT

Door het gebruik van Model View Presenter is het mogelijk om functionaliteit volledig te herbruiken. De classes die nodig zijn om XML bestanden te schrijven en in te lezen worden in zowel de Viewer als in de Extractor gebruikt.
Er is gekozen om voor beide applicaties verschillende View implementaties te ontwikkelen, zodat de logica volledig van de user interface gescheiden is.
Door de zojuist genoemde functionaliteit in beide applicaties te gebruiken, wordt meteen de herbruikbaarheid van MVP getoond.

2.4.1 XML IMPORTEREN

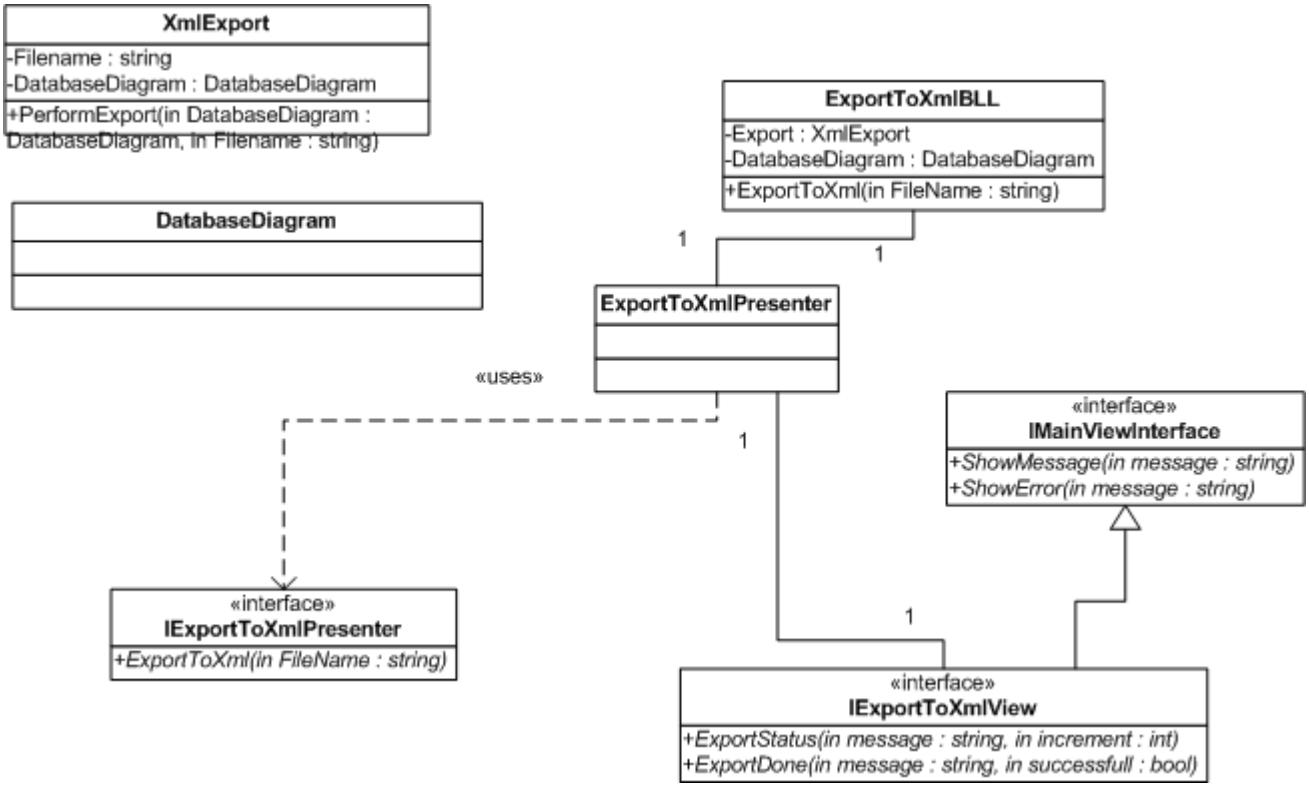
Het importeren van een XML-bestand wordt momenteel alleen gebruikt in de Viewer applicatie. Toekomstig onderhoud kan wellicht deze functionaliteit nodig hebben in de Extractor applicatie, waardoor deze functionaliteit onder het gedeelde deel van het systeem valt.
De View-implementatie is niet in dit diagram verwerkt omdat elke applicatie binnen het systeem een verschillende implementatie van de View kan hebben.



Afbeelding 5, Klassendiagram van de functionaliteit waarbij een XML-bestand geïmporteerd wordt.

2.4.2 EXPORTEREN NAAR XML

Het exporteren naar een XML-bestand heeft enkel een ingeladen instantie van de DatabaseDiagram class nodig. Aangezien zowel de XML-importeer-actie als de database importeer-actie een DatabaseDiagram instantie aanmaken, is deze functionaliteit zonder enige aanpassing in zowel de Viewer als de Extractor te gebruiken. In afbeelding 6 is geen View-implementatie gedocumenteerd, dit omdat beide applicaties binnen het systeem verschillende implementaties van de View hebben.



Afbeelding 6, Klassendiagram van de functionaliteit waarbij data naar een XML-bestand geëxporteerd wordt.

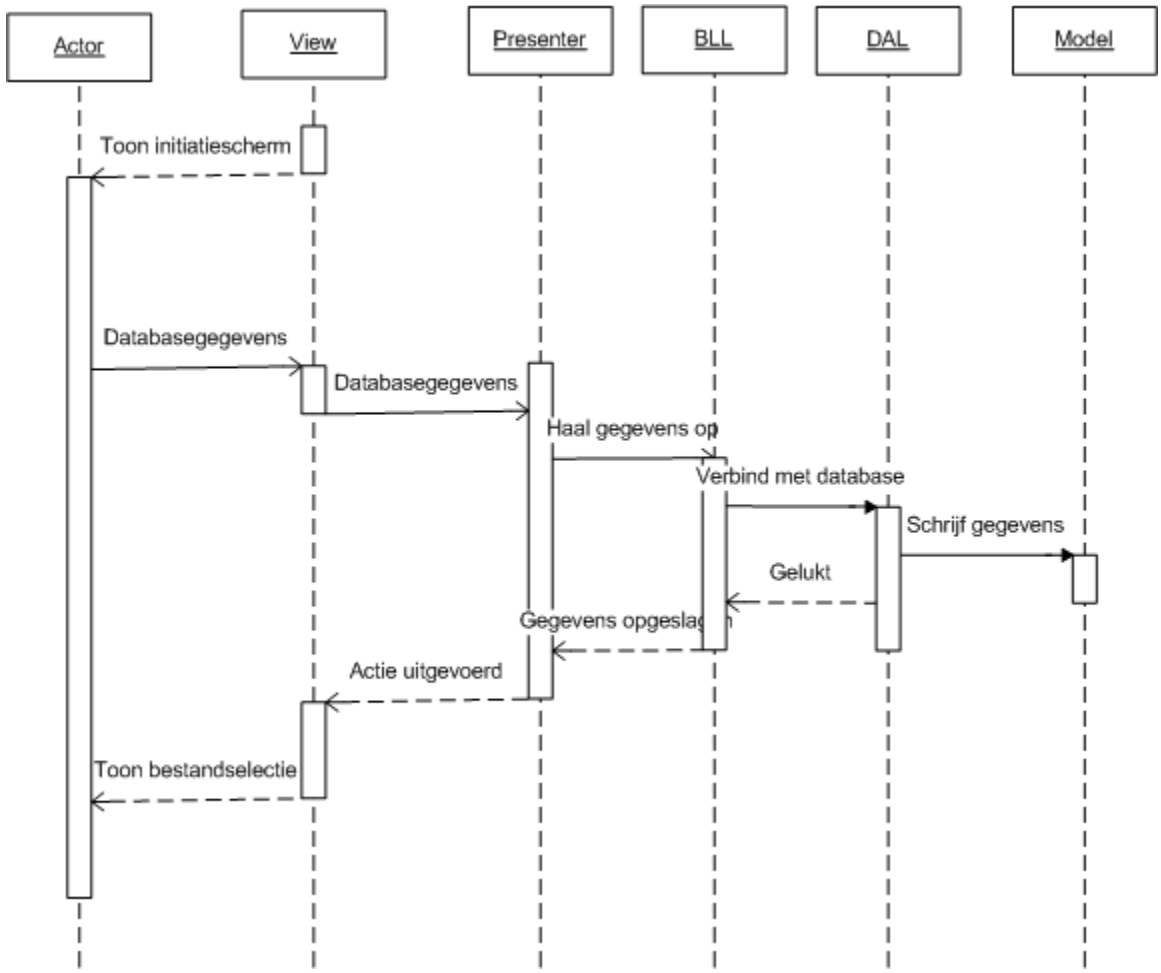
3. SEQUENCEDIAGRAMMEN

Dit hoofdstuk toont de sequencediagrammen voor het te realiseren systeem. De sequencediagrammen tonen de mogelijke acties van de gebruiker en de reactie van het systeem.

3.1 EXTRACTOR

UITLEG

Afbeelding 7 toont het gedrag van de Extractor applicatie wanneer een gebruiker met een database wilt verbinden en vervolgens een XML export van de structuur van deze database wilt maken.



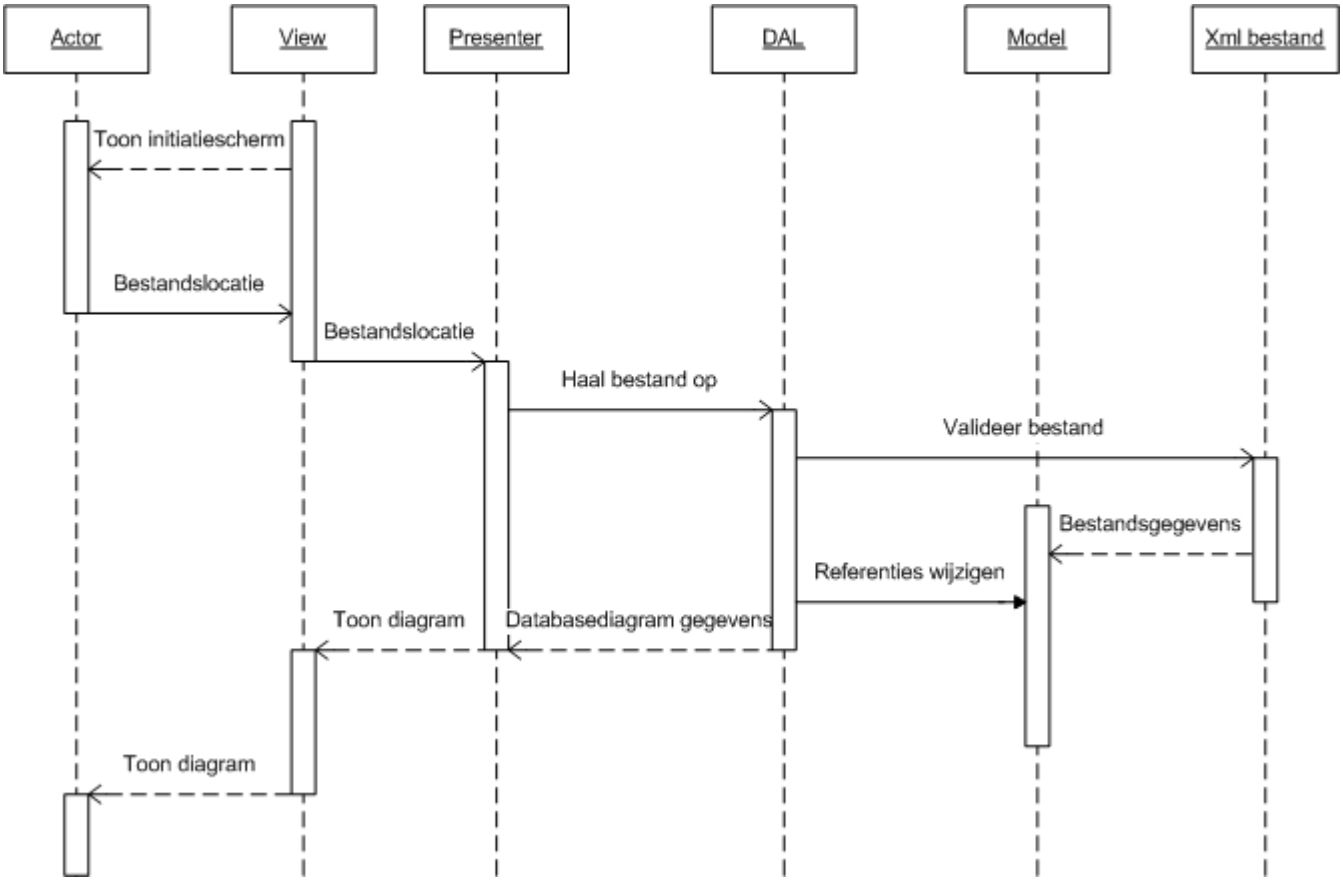
Afbeelding 7, Sequence diagram van de functionaliteit waarbij data naar een XML-bestand geëxporteerd wordt.

3.2 VIEWER

3.2.1 XML BESTAND IMPORTEREN

UITLEG

De Actor importeert een XML bestand waar data in staat wat een databasestructuur beschrijft.
De applicatie toont vervolgens het databasediagram.

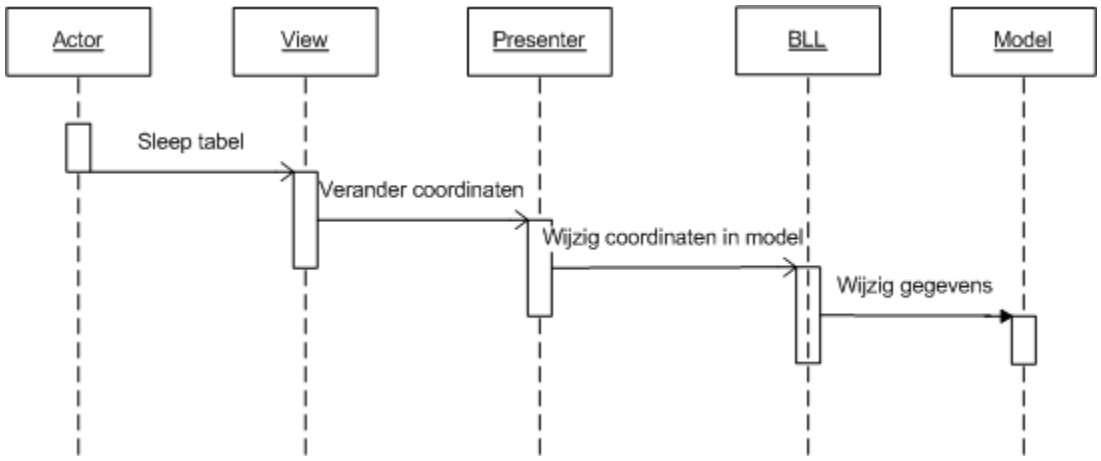


Afbeelding 8, Sequence diagram van de functionaliteit waarbij een XML-bestand geïmporteerd wordt.

3.2.2 INDELING DATABASEDIAGRAM WIJZIGEN

UITLEG

De Actor verplaatst een tabel in het databasediagram door middel van slepen met de muis.

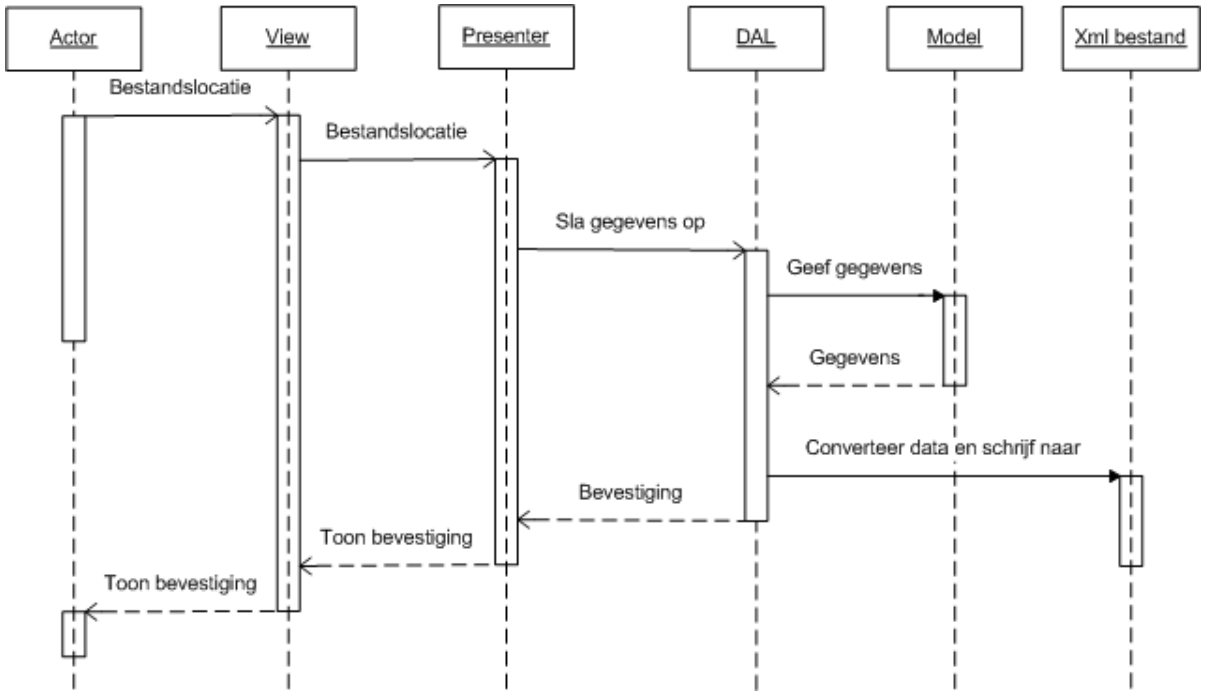


Afbeelding 9, Sequence diagram van de functionaliteit waarbij de indeling van een databasediagram gewijzigd wordt.

3.2.3 XML BESTAND EXPORTEREN

UITLEG

De Actor heeft een databasediagram op het scherm staan en wenst deze te exporteren naar een XML bestand.



Afbeelding 10, Sequence diagram van de functionaliteit waarbij data naar een XML-bestand geëxporteerd wordt.

4. ITERATIE: OVERZICHTELIJKHEID

4.1 ARCHITECTUUR

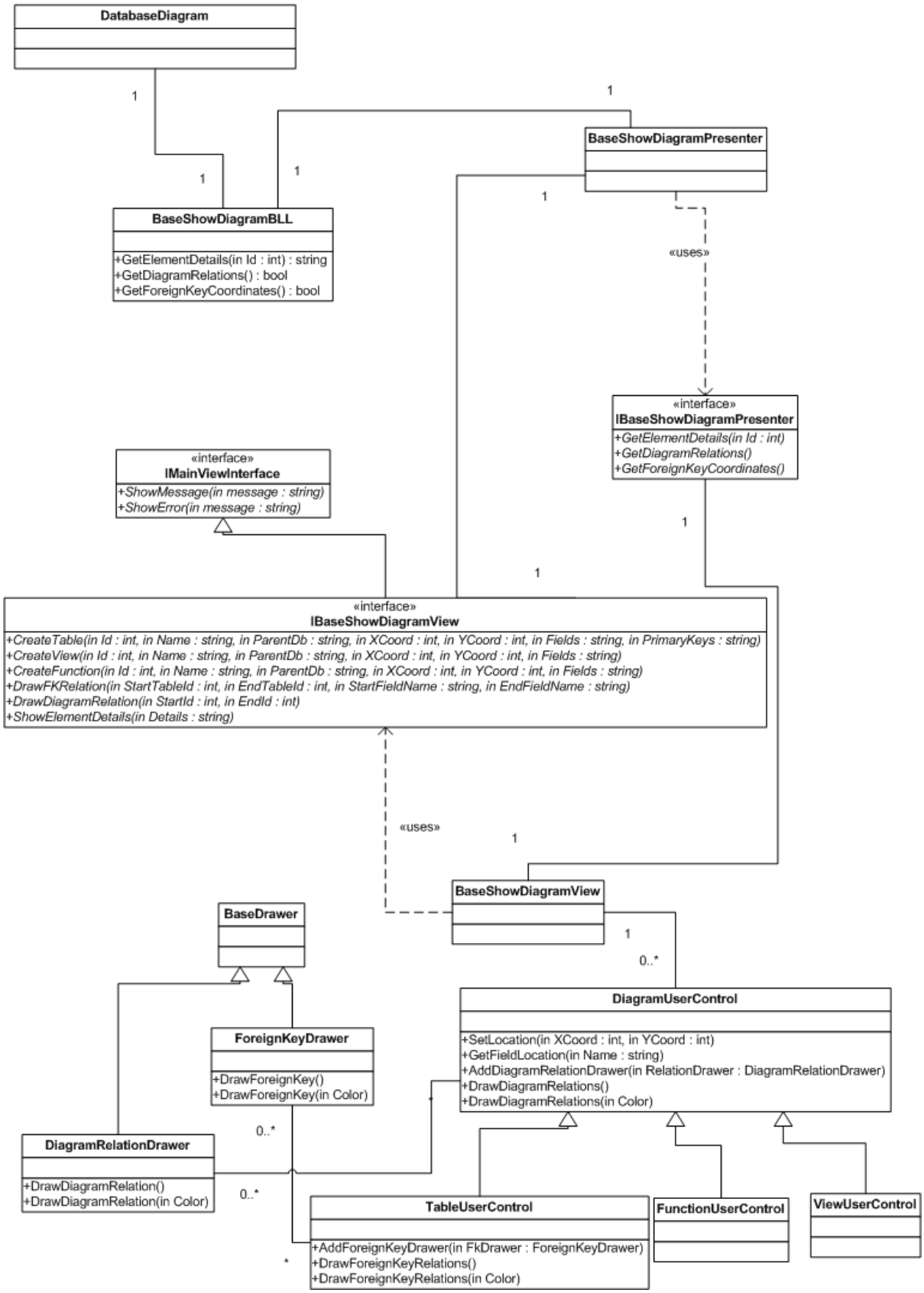
De in de "overzichtelijkheid"-iteratie te realiseren functionaliteit maakt gebruik van dezelfde architectuur als reeds geïmplementeerd.

4.2 KLASSENDIAGRAMMEN

De implementatie van de "overzichtelijkheid"-iteratie voegt extra functionaliteit toe, maar moet ook functionaliteit ondersteunen die op dit moment al in het systeem geïmplementeerd is. Om codeduplicatie te voorkomen is ervoor gekozen om classes toe te voegen die de basisfunctionaliteit voor het tonen van een database diagram ondersteunen.
De functionaliteit waarbij het complete diagram getoond wordt en de functionaliteit die in de "overzichtelijkheid" iteratie gerealiseerd wordt breiden beide deze classes uit met hun eigen implementatie.

4.2.1 CLASSES VOOR BASISFUNCTIONALITEIT DIAGRAM TONEN

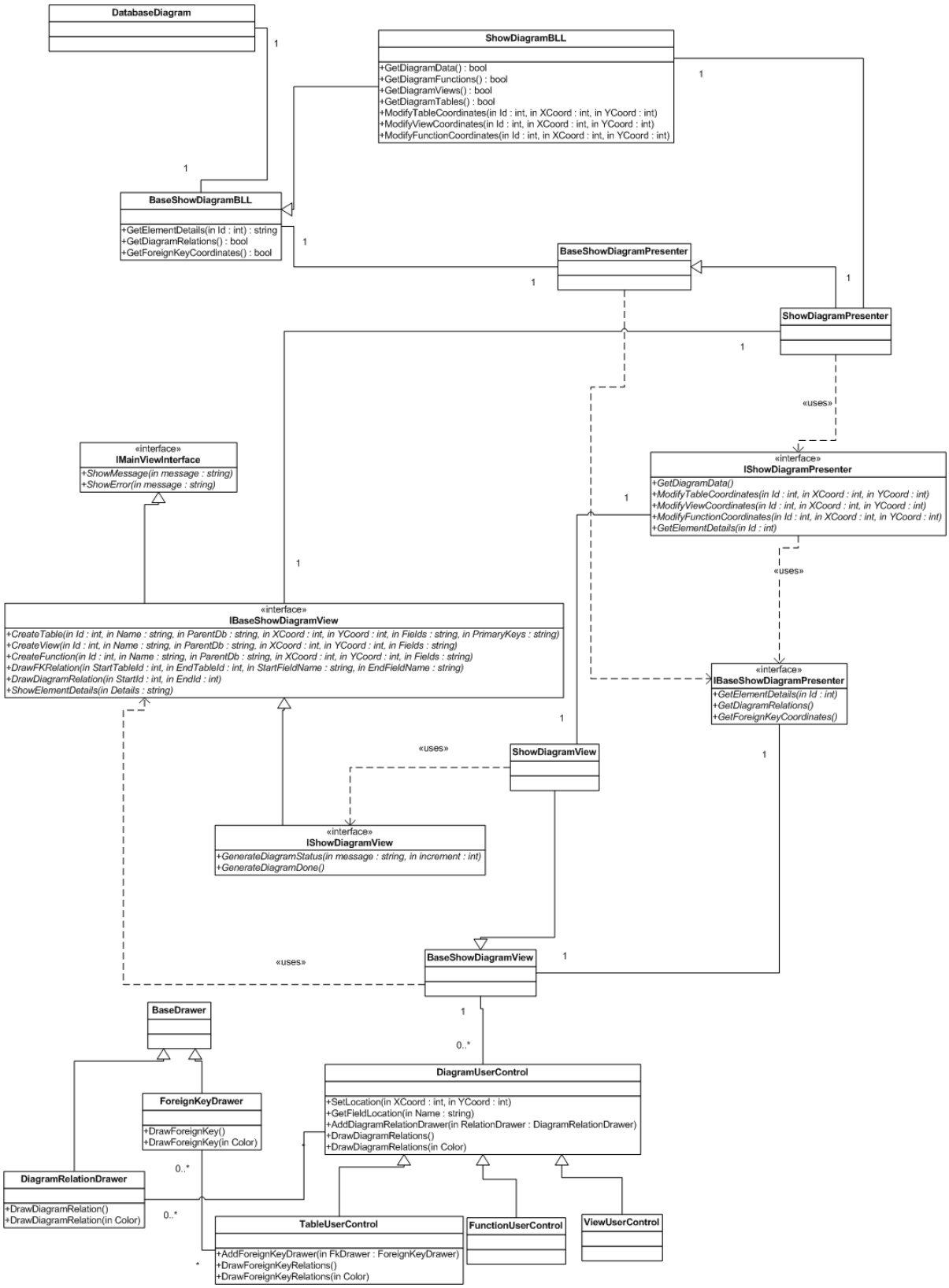
Afbeelding 11 toont de classes benodigd voor de basisfunctionaliteit van het tonen van een database diagram op het scherm. Het slepen van elementen wordt ook ondersteund in de basisclasses, maar de gewijzigde locatie wordt niet opgeslagen in het Model.
De basisclasses ondersteunen het ophalen van gedetailleerde informatie van één element door de GetElementDetails methode.



Afbeelding 11, Klassendiagram van de basisfunctionaliteit voor het tonen van databasediagrammen.

4.2.2 CLASSES VOOR TONEN COMPLEET DIAGRAM

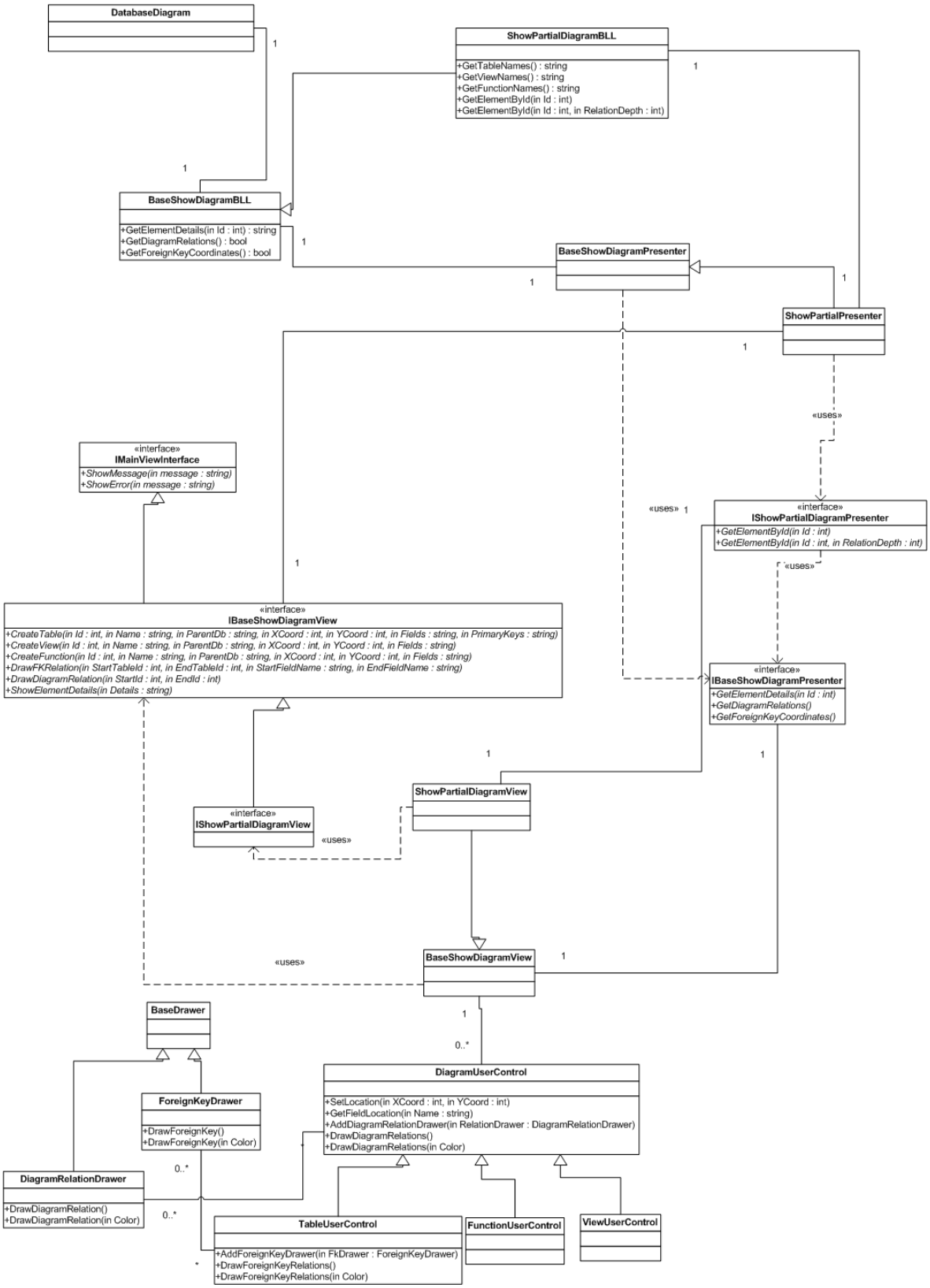
In afbeelding 12 worden alle classes getoond die benodigd zijn voor het tonen van het complete database diagram. De classes benodigd voor de basisfunctionaliteit van een database diagram worden ook getoond.
In afbeelding 12 wordt het ondersteund om een compleet database diagram op te vragen en de gewijzigde locatie na het slepen van een element op te slaan in het Model.



Afbeelding 12, Klassendiagram van de functionaliteit om complete databasediagrammen te tonen.

4.2.3 CLASSES VOOR IMPLEMENTATIE USE CASES "OVERZICHTELIJKHEID"-ITERATIE

Afbeelding 13 toont de classes die nodig zijn voor de implementatie van de functionaliteit van de "overzichtelijkheid" iteratie. De classes benodigd voor de basisfunctionaliteit van het tonen van een database diagram worden ook weergegeven in afbeelding 13. Afbeelding 13 biedt ondersteuning voor het toevoegen van een enkel element aan het weergegeven database diagram en het toevoegen van een element met de elementen waarvan/waarnaar een directe verwijzing is.



Afbeelding 13, Klassendiagram van de functionaliteit die toegevoegd wordt in de "Overzichtelijkheid"-iteratie.

Construction rapport

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

INHOUDSOPGAVE

INLEIDING

Dit document is geschreven naar aanleiding van het afstudeerproject bij Centric. Tijdens het afstudeerproject wordt een softwaresysteem ontwikkeld waarmee de metadata van Oracle databases als databasediagram op het scherm getoond kan worden.

Dit document beschrijft de bevindingen tijdens de Construction fase.
Dit document bevat de requirements, use cases en het domeinmodel die opgesteld zijn aan de hand van de eisen van de opdrachtgever. Dit document wordt gebruikt om aan te tonen welke functionaliteit geïmplementeerd is in de applicaties die in de construction fase gebouwd zijn.

1. BOUWEN VAN APPLICATIES

1.1 EXTRACTOR

1.1.1 UITBREIDINGSMOGELIJKHEID

Het doel van de Extractor is het converteren van data uit een Database Management Systeem (DBMS) naar een ander formaat, zoals een XML-bestand.
Met bovenstaand doel voor ogen staat het Model gedeelte van Model View Presenter (MVP) los van elke importeer functionaliteit van de Extractor. Doordat de data losstaat van de importeerlogica is het mogelijk om nieuwe importeer functionaliteit toe te voegen zonder de bestaande source code aan te passen, bijvoorbeeld de mogelijkheid om data vanuit een MySQL database of een sql bestand uit te kunnen lezen.

Tijdens het ontwikkelen van de Extractor is de importeer functionaliteit vanuit een Oracle database en de exporteer functionaliteit naar een XML bestand ontwikkeld.

1.2 VIEWER

1.2.1 UITBREIDINGSMOGELIJKHEID

Het doel van de Viewer is om een diagram te genereren met data vanuit een gegevensbron.
Net als bij de Extractor staat de functionaliteit van de Viewer compleet los van de importeer en exporteer functionaliteit.

Tijdens het ontwikkelen van de Viewer is het importeren van- en het exporteren naar een Xml bestand geïmplementeerd.

1.2.2 PLAATSBEPALEN VAN ELEMENTEN

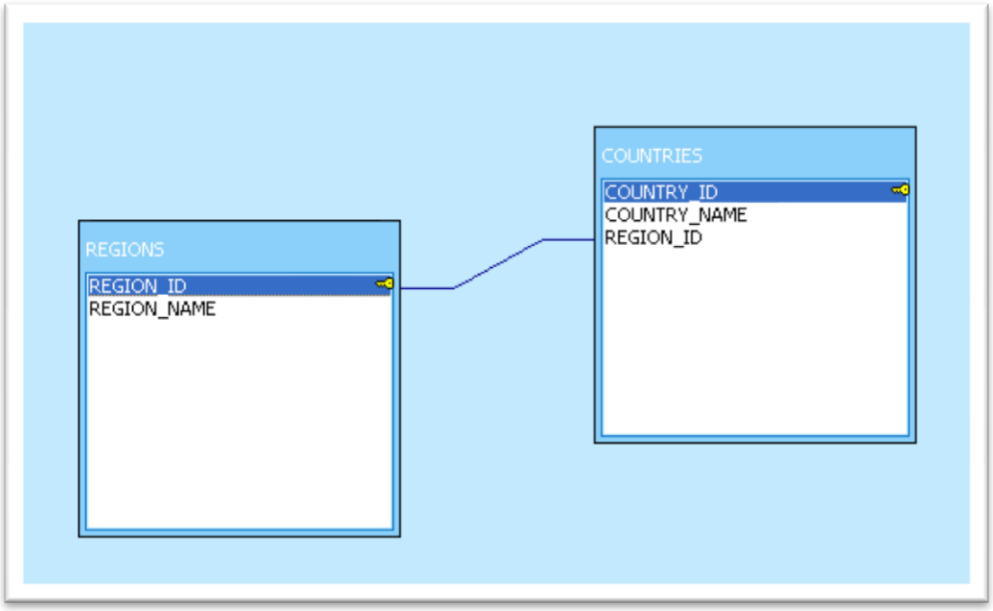
Om een overzichtelijk diagram te genereren is het noodzakelijk om ervoor te zorgen dat elementen binnen het diagram overzichtelijk op het scherm komen te staan.
De Viewer zorgt ervoor dat de elementen binnen het diagram in rijen onder elkaar komen te staan. Doordat rijen van elementen ontstaan, wordt een relatief overzichtelijk diagram getoond.

1.2.3 LIJNCOÖRDINATEN BEPALEN

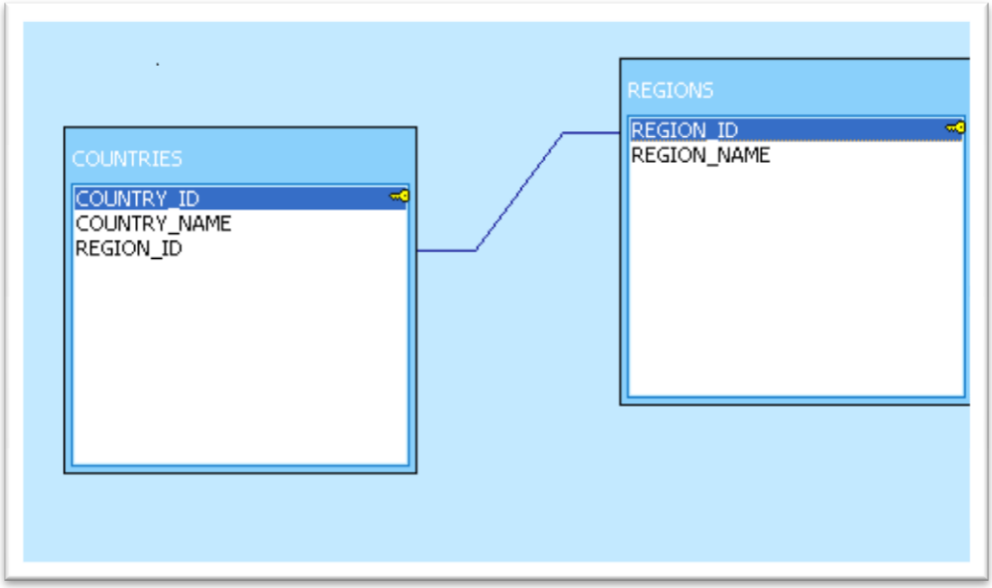
DYNAMISCHE COÖRDINATEN

Wanneer het databasediagram zich op het scherm bevindt, is het mogelijk voor de gebruiker om de elementen binnen het diagram te slepen. Wanneer een sleepactie voltooid is, berekend de Viewer nieuwe coördinaten voor de lijnen die bij het gesleepte element horen.

De coördinaten van de lijnen worden berekent aan de hand van de posities van de gerelateerde diagram elementen. De lijnen kunnen enkel aan de linker- of rechterzijde van het element aangrijpen. De Viewer berekent welke zijdes het dichtst bij elkaar staan om zo de meest efficiënte lijn te tekenen. Zie afbeelding 1 voor een voorbeeld.



Afbeelding 1, Twee tabellen waarvan één een vreemde sleutel verwijzing naar de ander heeft.



Afbeelding 2, Dezelfde tabellen als hierboven, maar nadat de gebruiker de tabel "Countries" versleept heeft en de Viewer de nieuwe coördinaten berekend heeft.

De coördinaten worden in het XML bestand opgenomen wanneer de gebruiker de XML exporteer functionaliteit gebruikt.

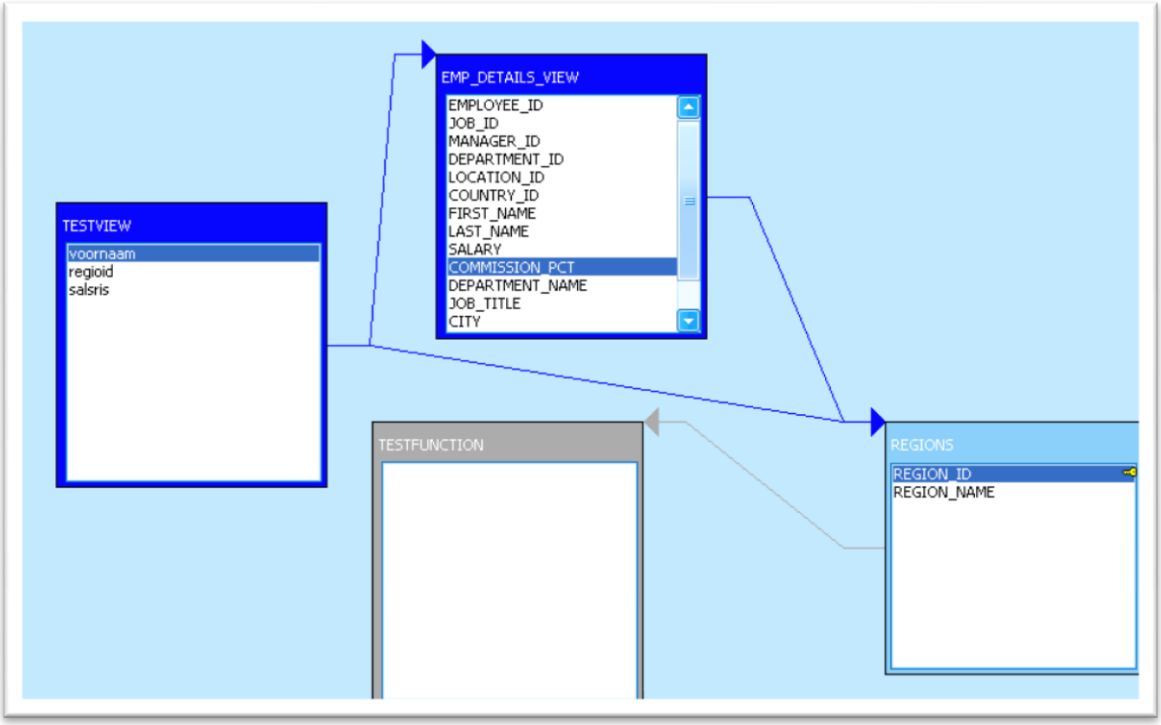
FOREIGN KEYS

Een wens van de opdrachtgever is dat de vreemde sleutels aangeduid worden met een lijn die direct van het vreemde sleutel veld naar het primaire sleutels veld verwijst. De locatie van de lijnen wordt bepaald aan de hand van de locatie van de vreemde sleutels en de primaire sleutels op de elementen binnen het diagram. Het systeem houdt rekening met de locatie en regelhoogte van de lijst waarin de sleutels zich bevinden.

Primaire sleutels worden aangegeven door een sleutel icoon.

DIAGRAMELEMENT REFERENTIES

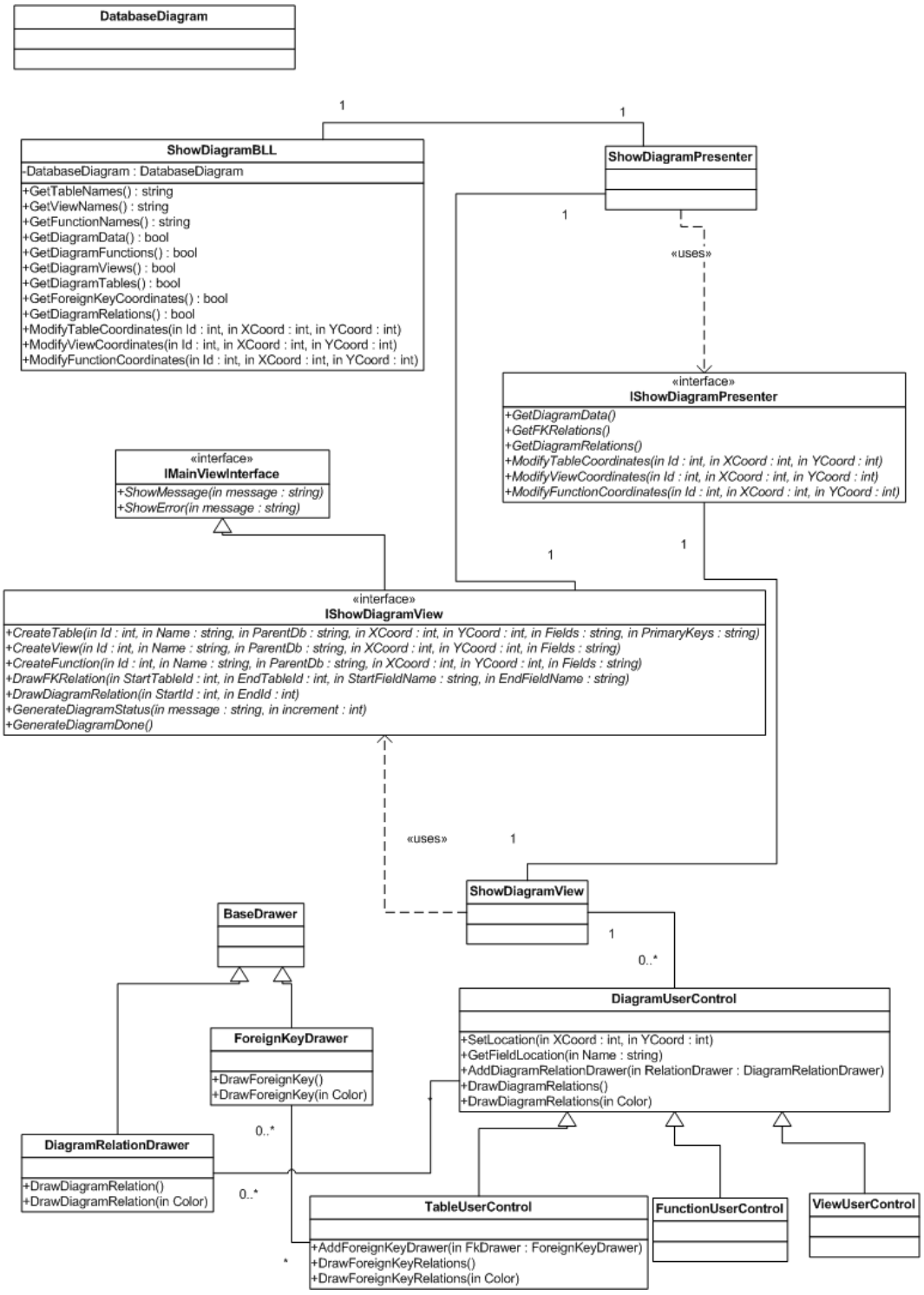
In een DBMS kunnen elementen zoals vreemde sleutels referenties hebben naar primaire sleutels, of refereren Views naar velden in een Table. De referenties tussen diagramelementen, zoals tabellen en views, worden aangeduid door een lijn met een pijl aan het eind. De lijn vertrekt vanuit het midden van het eerste element en eindigt aan de bovenkant van het element waarnaar verwezen wordt. Blauwe lijnen staan voor relaties vanuit een View, grijze lijnen staan voor relaties naar een functie. Zie afbeelding 3 voor een voorbeeld.



Afbeelding 3, Verschillende relaties in de Viewer.

1.2.4 TONEN COMPLEET DIAGRAM

Het tonen van een diagram in de Viewer, waarmee de locatie van de diagramelementen gewijzigd kan worden, vergt enige logica in de View. Tijdens de implementatie van deze use case zijn UserControls toegevoegd waarmee lijnen getekend kunnen worden of gesleept kunnen worden.
Afbeelding 4 toont de structuur van deze use case, inclusief de UserControls.



Afbeelding 4, Classdiagram voor het tonen van complete databasediagrammen.

OPHALEN VAN COMPLEET DIAGRAM

Wanneer ShowDiagramView getoond wordt, wordt direct de gegevens van het te tonen diagram opgehaald. Het ophalen van de gegevens wordt gedaan via een BackgroundWorker class, deze class zorgt ervoor dat een Thread[1] op de achtergrond wordt gestart en de gegevens asynchroon opgehaald worden, waardoor de gebruikersinterface niet vastloopt en feedback over de progressie van het proces mogelijk is.
ShowDiagramPresenter kan tijdens het doorlopen van de gegevens voor het diagram voor elk element een aparte Create-methode (zoals CreateTable in afbeelding 4) aanroepen in ShowDiagramView.
Een bijkomend voordeel van het aanroepen van een Create-methode als CreateTable is dat geen ingewikkeld dataformaat gebruikt hoeft te worden.

USERCONTROLS

De TableUserControl, FunctionUserControl en ViewUserControl overerven van de DiagramUserControl, welke de basis Usercontrol is. De DiagramUserControl bevat de methodes om sleep acties af te vangen en om gegevens weer te geven.

DRAWER CLASSES

De BaseDrawer class is de basisclass die het berekenen van de kortste route tussen twee DiagramUserControl instanties en bevat basis attributen benodigd voor het tekenen van een lijn.
De class ForeignKeyDrawer bevat methodes die ervoor zorgen dat de vreemdesleutelrelaties getekend worden.
De DiagramRelationDrawer bevat functionaliteit om relaties tussen tabellen, views en functies te tekenen op het scherm.
Zowel de ForeignKeyDrawer als de DiagramRelationDrawer hebben een methode waarmee met een verschillende kleur getekend kan worden, waardoor in de toekomst individuele lijnen duidelijker weergegeven kunnen worden.

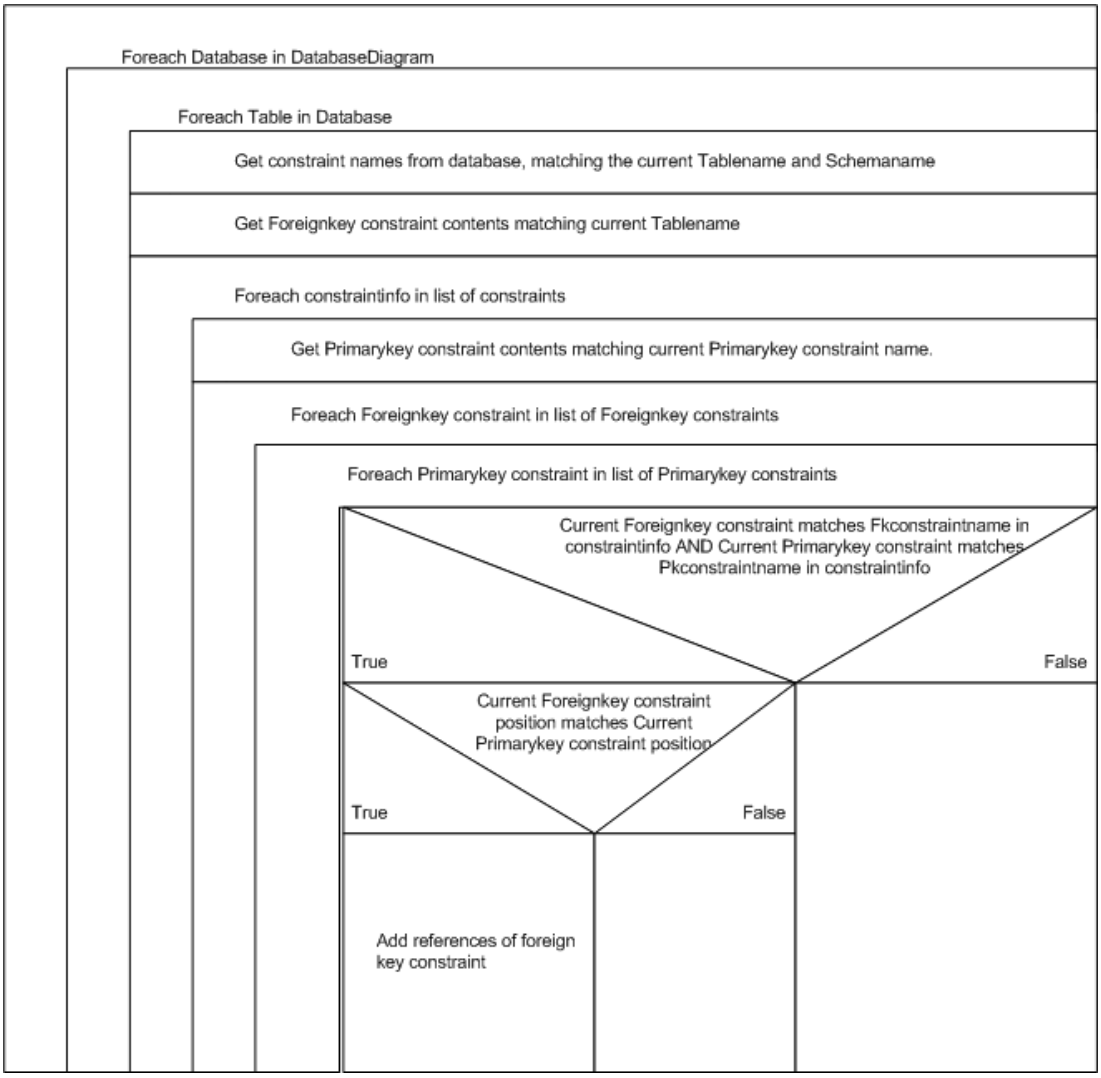
[1] Een Thread is een proces dat binnen een proces uitgevoerd wordt, waardoor er van meerdere processors gebruik gemaakt kan worden.

3. PROBLEMEN

3.1 ORACLE IMPORT

Tijdens het ontwikkelen van de Oracle importeer functionaliteit is gebruik gemaakt van het DotConnect[2] component. Door het gebruik van DotConnect is het mogelijk dat gebruikers, zonder de Oracle client te hoeven installeren, verbinding kunnen maken met een Oracle DBMS. Het is echter wel noodzakelijk om de DotConnect client te installeren om gebruik van de Extractor te maken.
De Oracle importeer functionaliteit zorgt ervoor dat data vanuit een Oracle database in het Model geladen wordt.

3.1.1 PROGRAMMA STRUCTUUR DIAGRAM



Afbeelding 5, PSD van het ophalen van data uit een Oracle DBMS.

[2] Software om te kunnen verbinden met een Oracle server, ontwikkeld door DevArt.

Afbeelding 5 beschrijft de methode die relaties van vreemde sleutels ophaalt uit een Oracle database. In Oracle verwijzen vreemde sleutels niet direct naar primaire sleutels, maar naar constraints. Een constraint kan meerdere kolommen omvatten aan zowel de vreemde sleutel kant als de primaire sleutel kant. De koppeling tussen constraints wordt geregeld in de User_constraints view van Oracle.
De User_cons_columns view bevat koppeling tussen kolomnaam en constraint-naam.

Om uit te vinden naar welke kolom een vreemde sleutel verwijst, moeten drie query's op de Oracle database worden uitgevoerd. Tevens moet over de gevonden resultaten heen geïtereerd worden.

3.1.2 STATISTIEKEN

Om een goed beeld te krijgen van het effect van het ophalen van vreemde sleutels op de totale export, is een test uitgevoerd waarbij de volledige testdatabase van Centric opgehaald wordt.
De testdatabase van Centric bestaat uit 29 schema's, die samen de volgende elementen bevatten:

- 2444 tabellen
- 17341 kolommen
- 234 views
- 28 functions

De export is opgebouwd uit verschillende delen. Eerst worden alle tabellen, views, functions en bijbehorende kolommen opgehaald.
Vervolgens worden de primaire sleutels opgehaald en als laatste worden vreemde sleutels opgehaald.

Actie	Tijdsduur tijdens ophalen testdatabase
Ophalen van tabellen	3 seconden
Ophalen van views	7 seconden
Ophalen van functions	8 seconden
Ophalen van kolommen	14 seconden
Ophalen van primaire sleutels	4 seconden
Ophalen van vreemde sleutels	1004 seconden

Uit bovenstaand tabel blijkt dat het ophalen van vreemde sleutels zo'n 96% van de tijd in beslag neemt.

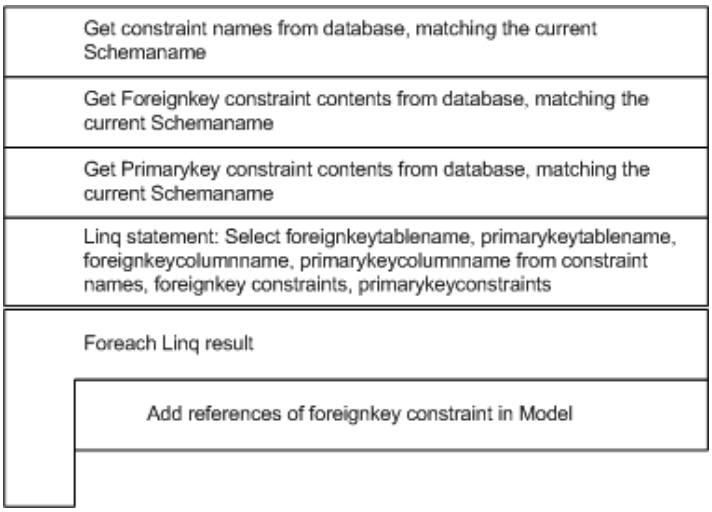
3.1.3 MOGELIJKE OPLOSSING

De reden waardoor het ophalen van de vreemde sleutels zo traag is, is omdat veel SQL query's naar de database gestuurd worden. Om de performance drastisch te verbeteren moet het aantal query's gereduceerd worden.
Tijdens het ontwikkelen van de Oracle export is een alternatief op deze trage manier van vreemde sleutels ophalen gebouwd. Het alternatief gebruikt drie query's per databaseschema, waarna de applicatie alle data met elkaar koppelt.
Door bovenstaand alternatief wordt de databaseserver minder belast, maar de computer waar de applicatie op uitgevoerd wordt, moet erg veel data verwerken.
De uiteindelijke snelheidswinst is door deze oplossing minimaal.

In het boek "Pro C# with .NET 3.0" geschreven door Andrew Troelsen wordt een techniek genaamd Language Integrated Query (LINQ) besproken waarmee vanuit C# verschillende type databronnen gecombineerd kunnen worden om gegevens te zoeken. In dit geval wordt LINQ gebruik om data uit verschillende DataTables met elkaar te koppelen en de juiste resultaten eruit te halen.
DotConnect wordt gebruikt als communicatiemiddel met de database, maar de resultaten vanuit de database worden doorzocht door middel van LINQ.
Onderstaande tabel laat een duidelijk overzicht zien van de tijdswinst.

Actie	Tijdsduur tijdens ophalen testdatabase zonder LINQ	Tijdsduur tijdens ophalen testdatabase met LINQ
Ophalen van tabellen	3 seconden	2 seconden
Ophalen van views	7 seconden	7 seconden
Ophalen van functions	8 seconden	9 seconden
Ophalen van kolommen	14 seconden	12 seconden
Ophalen van primaire sleutels	4 seconden	4 seconden
Ophalen van vreemde sleutels	1004 seconden	120 seconden

Door LINQ toe te passen is de tijdsduur van het ophalen van de testdatabase ongeveer 700% korter geworden.
Naast de snelheidswinst is de methode om vreemde sleutels op te halen een stuk eenvoudiger geworden door het gebruik van LINQ. Afbeelding 6 toont het PSD van de methode met het gebruik van LINQ.



Afbeelding 6, PSD van het ophalen van data uit een Oracle DBMS met gebruik van LINQ.

3.2 TONEN VAN GROTE DATABASES

Het is niet mogelijk om de volledige testdatabase van Centric te tonen.
Wanneer de testdatabase op het scherm getoond wordt, is het aantal elementen op het scherm te veel voor .NET en treed een fatale fout op.
In .NET kunnen standaard zo'n 10.000 Control-handles aangemaakt worden. Wanneer een erg groot diagram getoond wordt, kan dit limiet overschreden worden waardoor .NET een fout geeft.

Testplan

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

INHOUDSOPGAVE

INLEIDING

Dit document beschrijft de testactiviteiten die uitgevoerd worden tijdens de afstudeerperiode.

De beschreven tests hebben betrekking op twee applicaties die samen een softwarepakket vormen. Het softwarepakket zal gebruikt worden om Oracle databases te kunnen exporteren naar een XML-document om hiermee vervolgens in de tweede applicatie, de Viewer, een databasediagram te genereren.

Het eerste hoofdstuk beschrijft kort het doel van de opdracht, waarna hoofdstuk twee een korte beschrijving geeft over het te testen systeem. De testactiviteiten worden in hoofdstuk drie beschreven.

1. OPDRACHT

1.1 OPDRACHTGEVER

De opdrachtgever is de technisch manager van de afdeling MVV bij Centric. Tijdens de ontwikkeling van het systeem en het testproject, is er wekelijks overleg met de opdrachtgever.

1.2 OPDRACHTNEMER

Het testplan, de testscripts en het testrapport worden door de ontwikkelaar van het systeem opgesteld. De systeemtests en de acceptatietests, met uitzondering van de performancetests, worden door het testteam van de afdeling MVV uitgevoerd.
De performancetests worden door de ontwikkelaar van het systeem uitgevoerd.

1.3 BESCHOUWINGSGEBIED

Tot het beschouwingsgebied van de acceptatietests en de systeemtests behoren:

- De release van de Viewer waarin de requirements, die in het inception rapport zijn vastgesteld, zijn geïmplementeerd
- De release van de Extractor waarin de requirements, die in het inception rapport zijn vastgesteld, zijn geïmplementeerd

1.4 DOEL

Bepaal in hoeverre het systeem voldoet aan de requirements uit het inception rapport. Geef aan welke systeemdelen niet correct functioneren, zodat de ontwikkelaar(s) het kunnen herstellen. Stel op basis van de bevindingen en de risico's een advies over het in productie nemen van het systeem.

1.5 RAPPORTAGE

De resultaten van het de uitgevoerde tests moeten in een testrapport vastgelegd worden.
Per bevinding moet het prioriteitniveau worden vastgesteld en gedocumenteerd worden.
Resultaten van de performancetests moeten met feiten beargumenteerd worden.
Tevens moet de status van de geteste (sub)systemen beargumenteerd worden en er moet een advies over de release van het desbetreffende (sub)systeem gegeven worden.

2. SYSTEEM

De belangrijkste functionaliteit van het systeem is het omzetten van een databasestructuur naar een databasediagram. Het systeem moet een databasestructuur uit kunnen lezen om deze vervolgens naar een XML-bestand weg te schrijven. Het weggeschreven XML-bestand wordt gebruikt om een visuele weergave van de databasestructuur te genereren.

Het te testen systeem is verdeeld in twee verschillende applicaties met elk hun eigen doel.
Beide applicaties gebruiken de Nederlandse taal in hun gebruikersinterface en implementeren de Centric stijlgids.
Het systeem staat los van andere systemen binnen Centric.

2.1 EXTRACTOR

De eerste applicatie is de Extractor, deze applicatie is gebouwd om metadata vanuit een Oracle database te halen en weg te schrijven naar een XML-bestand. De Extractor maakt gebruik van DotConnect om met een Oracle database te verbinden. Om te kunnen verbinden met Oracle is het noodzakelijk om de naam, locatie, gebruikersnaam en wachtwoord van de database in te voeren.

2.2 VIEWER

De tweede applicatie is de Viewer, deze applicatie heeft als doel om databasediagrammen te tonen aan de hand van data die door de Extractor geëxporteerd is. De databasediagrammen die op het scherm getoond worden zijn door de gebruiker in te delen door elementen binnen het diagram van locatie te veranderen. De Viewer ondersteund de mogelijkheid om het opnieuw ingedeelde diagram te exporteren naar XML-bestand, waardoor de indeling bewaard blijft.

3. TESTBASIS

De testbasis voor het opstellen van het testplan bestaat uit de requirements en de use cases zoals gedocumenteerd in het inception rapport en het plan van aanpak. De testbasis wordt gebruikt om het testplan op te stellen en de specifieke tests te specificeren.

4. TEST-STRATEGIE

In deze teststrategie is ervan uitgegaan dat het ontwikkelteam de programmatests, in de vorm van unit-tests heeft uitgevoerd. Het testteam zal uitsluitend systeemtests en acceptatietests uitvoeren.

De teststrategie koppelt het belang dat aan deelsystemen gehecht wordt en de uit te voeren tests.

4.1 KWALITEITSATTRIBUTEN

In overleg met de opdrachtgever zijn de kwaliteitsattributen vastgesteld samen met hun relatieve belang. Het relatieve belang bepaald de verdeling van de testinspanning voor het te testen systeem.

Kwaliteitsattributen	Relatief belang
Beveiliging	5
Controleerbaarheid	15
Functionaliteit	50
Gebruikersvriendelijkheid	10
Inpasbaarheid	0
Performance	20
Totaal in %	**100**

4.2 STRATEGIEMATRIX

Zoals in hoofdstuk 2 is beschreven, bestaat het systeem in twee delen. Het systeem staat los van andere systemen bij Centric.

in overleg met de opdrachtgever is per deelsysteem bepaald wat het relatieve belang is binnen het totale systeem.
Een + in de matrix geeft aan dat het kwaliteitsattribuut in het deelsysteem aanwezig is.
Een ++ in de matrix geeft aan dat er relatief veel aandacht aan het kwaliteitsattribuut gegeven wordt.

Onderstaande tabel is de strategiematrix voor de acceptatietest van de Extractor en de Viewer.

	Extractor	Viewer	Relatief belang %
Beveiliging	+		**5**
Controleerbaarheid	++	+	**15**
Functionaliteit	++	++	**50**
Gebruikersvriendelijkheid		+	**10**
Inpasbaarheid			**0**
Performance	+	++	**20**
Relatief belang in %	**45**	**55**	**100**

5. TESTACTIVITEITEN

5.1 SYSTEEMTESTS

De systeemtest zal opgedeeld zijn in verschillende (sub)tests. In de volgende paragrafen zal uitgelegd worden welke (sub)tests er uitgevoerd worden.

5.1.1 SCHERMTEST

De schermtest heeft als doel om te controleren of de schermen foutloos werken. Wanneer de schermtest succesvol is afgerond, bevat het systeem geen fouten tijdens het laden van de schermen.

EXTRACTOR

Uitgangspunten:

- Er is een testdatabase beschikbaar met minimaal één tabel
- Elke test heeft betrekking op één scherm

Navigatie tussen schermen:

- Na het opstarten van de applicatie wordt het scherm getoond waar databaseconnectie gegevens ingevoerd kunnen worden
- De knop "Verbinden" start het proces dat data uit de Oracle database ophaalt. Wanneer deze gereed is, schakelt de knop "Ok" naar het bestandsselectie scherm.
- De gebruiker kan tussen schermen schakelen door middel van de navigatiebalk aan de linkerzijde van de applicatie

VIEWER

Uitgangspunten:

- Er is een XML-document die, volgens de gedocumenteerde structuur, een databasestructuur bevat met minimaal één tabel
- Elke test heeft betrekking op één scherm

Navigatie tussen schermen:

- Na het opstarten van de applicatie wordt het XML-importeerscherm getoond.
- Wanneer er een bestand geselecteerd is en op de "importeer" knop gedrukt wordt het bestand geïmporteerd.
- Na het importeren wordt een scherm getoond waar de gebruiker een diagram samen kan stellen
- De gebruiker kan tussen schermen schakelen door middel van de navigatiebalk aan de linkerzijde van de applicatie

5.1.2 CONSISTENTIETEST

De consistentietest heeft als doel om te testen of de verschillende schermen dezelfde globale lay-out bevatten. Deze lay-out is door Centric van tevoren in de stijlgids vastgelegd. Het systeem bevat consistente schermen wanneer deze test succesvol is afgerond.

Uitgangspunten Extractor:

- Een stijlgids met de Centric lay-out regels is beschikbaar
- Alle termen in de user interface zijn in het Nederlands

Uitgangspunten Viewer:

- Een stijlgids met de Centric lay-out regels is beschikbaar
- Alle termen in de user interface zijn in het Nederlands

5.2 ACCEPTATIETESTS

De acceptatietest wordt uitgevoerd om te testen of de eisen van de opdrachtgever in de applicaties verwerkt zijn. De onderstaande acceptatietests worden uitgevoerd.

5.2.1 FUNCTIONELE ACCEPTATIETEST

Er wordt een functionele acceptatietest uitgevoerd. Deze test wordt uitgevoerd door middel van een van te voren gedefinieerd testscript wat doorlopen moet worden. Het testscript wordt vastgelegd in een template dat gebruikt wordt door testers bij Centric. De testscripts beschrijven stap voor stap wat de tester uit moet voeren om het systeem te testen.
De testscripts worden als externe documenten opgeleverd.

5.2.2 PERFORMANCETEST

Om de performance van de Extractor en de Viewer te testen, wordt er gebruik gemaakt van een databaseserver die zich op het interne netwerk van Centric bevindt. Voor deze test wordt de TestMV database gebruikt. De TestMV database is een testdatabase die door Centric regelmatig gebruikt wordt om te testen.

EXTRACTOR

IMPORTEREN VAN DATABASESTRUCTUUR

De performance van het exporteren van een databasestructuur hangt sterk af van een aantal factoren. Er zijn zo'n vier factoren te definiëren die impact hebben op de performance van het exporteren van een databasestructuur.

1. De rekenkracht van de client computer
2. De snelheid van het netwerk waar zowel de client computer als de database server zich op bevinden
3. De (beschikbare) rekenkracht van de database server
4. De complexiteit van een database

Op geen van deze factoren heeft de Extractor applicatie invloed. Dit zijn de externe factoren.

EXPORTEREN NAAR XML BESTAND

De geïmporteerde data vanuit een databron kan weggeschreven worden naar een XML bestand. Het exporteren van deze data kan een intensieve taak zijn, omdat kruisverwijzingen naar objecten omgezet moeten worden naar verwijzingen door middel van ID.

PERFORMANCE CRITERIA

De tijd dat de Extractor erover mag doen om de totale TestMV database uit te lezen is 5 minuten. Vervolgens mag het maximaal 30 seconden duren om de databasestructuur te exporteren naar een XML-bestand.

VIEWER

IMPORTEREN VAN XML BESTAND

De Viewer biedt de mogelijkheid om een XML bestand in te lezen, zodat er een databasediagram gegenereerd kan worden aan de hand van data die in de Extractor geëxporteerd is. Tijdens het importeren van een XML bestand moeten de verwijzingen door middel van ID omgezet worden naar verwijzingen naar objecten.

GENEREREN VAN DATABASEDIAGRAM

Het inlezen van een databasediagram kan een complexe en lange procedure zijn. De lengte van deze procedure hangt sterk af van het aantal tabellen en relaties in de database. Naarmate een database meer tabellen en relaties heeft, zal de tijd om het databasediagram op te bouwen groter worden.
De gebruiker verwacht geen laadtijden van meerdere minuten bij een database van ongeveer vijftig tabellen en honderd relaties.

SLEPEN VAN TABELLEN

De Viewer moet het afhandelen van sleepacties van de gebruiker doen in een acceptabele tijd. Bij het werken met het databasediagram moet de applicatie snel reageren op acties van de gebruiker, zodat er geen wachttijden optreden.

Wanneer de gebruiker een tabel verplaatst, moet de applicatie deze actie snel kunnen verwerken. De gebruiker moet niet belemmerd worden door lange wachttijden om coördinaten te berekenen.

PERFORMANCE CRITERIA

Het importeren van een XML bestand met de volledige TestMV database mag, net als het exporteren naar een XML bestand, niet meer dan 30 seconden duren.

Het genereren van een databasediagram mag niet meer dan 3 minuten duren wanneer de gehele TestMV database geïmporteerd wordt.

Het slepen van tabellen moet vloeiend en zonder wachttijden gaan. De gebruiker moet een vloeiende gebruikersbeleving hebben wanneer er gewerkt wordt met de ingeladen diagrammen.

5.3 TESTAUTOMATISERING

Zowel de Extractor als de Viewer applicatie wordt door middel van Unit-tests automatisch getest. De Unit-tests testen de geïmplementeerde functionaliteit op methode-niveau.
De Unit-tests worden uitgevoerd binnen Visual Studio en zijn onafhankelijk van het systeem waarop het wordt uitgevoerd.
Voordat een ontwikkelaar een wijziging op het versiebeheersysteem zet, worden de Unit-tests uitgevoerd om te controleren of de huidige versie correct functioneert.

6. TESTPLANNING

6.1 SYSTEEMTESTS

Bij elke release van zowel de Viewer als de Extractor moeten systeemtests uitgevoerd worden. Op het moment dat de applicaties niet voldoen aan de eisen, moeten de bevindingen genoteerd worden in het testrapport. Gewijzigde en toegevoegde functionaliteit heeft meer prioriteit dan ongewijzigde functionaliteit, maar regressietests moeten wel uitgevoerd worden.

6.2 ACCEPTATIETESTS

Alle acceptatietests moeten uitgevoerd worden wanneer een release zich in de acceptatiefase bevindt. Elke functionaliteit waarbij minimaal een hoog risico aan vast zit, moet uitgebreid getest worden en elke bevinding moet opgelost worden voordat de release opgeleverd kan worden.

6.3 UNIT-TESTS

Voordat een versie van het systeem in het versiebeheersysteem gezet wordt, moeten alle Unit-tests worden uitgevoerd. Zodra een Unit-test niet succesvol is uitgevoerd, mag de versie niet in het versiebeheersysteem gezet worden.

Testrisico-analyse

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

INHOUDSOPGAVE

INLEIDING

Deze testrisico-analyse heeft betrekking op het systeem dat ontwikkeld is tijdens de afstudeerperiode van Ted de Koning. Dit document beschrijft kort wat de functionaliteit inhoudt, het risiconiveau dat bepaald is en de reden waarom het risiconiveau toegekend is aan de functionaliteit.

1. RISICO'S PER APPLICATIE

Dit hoofdstuk bevat de risicoanalyses per functionaliteit. De hoogte van het risico wordt aangeduid op verschillende niveau's.
De hoogte van het risico is afhankelijk van het risico dat de functionaliteit op het systeem heeft, maar ook het risico dat de functionaliteit op de omgeving, waar het systeem op uitgevoerd wordt, heeft.
Risicofactoren om het risiconiveau van een functionaliteit te bepalen zijn;

- Belang binnen het systeem
- Belang buiten het systeem
- Mogelijkheid tot muteren van data
- Complexiteit van de functionaliteit

Deze risicoanalyse houdt geen rekening met het belang van het systeem binnen de bedrijfsorganisatie, omdat dit belang op het moment van schrijven minimaal is.

1.1 EXTRACTOR

1.1.1 DATABASESTRUCTUUR INLADEN

Het inladen van een databasestructuur is een essentieel gedeelte van de Extractor applicatie.
De functionaliteit om een databasestructuur vanuit een Oracle database in het geheugen te laden maakt gebruik van DotConnect om met een Oracle database te verbinden. Om de gegevens op te halen wordt gebruik gemaakt van de GetSchema methode van DotConnect, deze methode voert selectie query's uit om metadata uit de database op te halen.
Deze functionaliteit heeft geen bestandstoegang op de computer waarop het uitgevoerd wordt en voert geen mutaties op de database uit. Het inladen van een databasestructuur in het Model is relatief complex.

CONCLUSIE

Doordat deze functionaliteit een essentieel gedeelte van de Extractor applicatie is, maar geen bestaande data kan muteren is het risico van deze functionaliteit hoog.

1.1.2 EXPORTEREN NAAR XML-BESTAND

Het exporteren van de ingeladen data naar een XML-bestand is, net als het inladen van de data, een essentieel gedeelte van de Extractor applicatie.
De functionaliteit heeft toegang tot het bestandssysteem van de computer waarop de applicatie wordt uitgevoerd en heeft de mogelijkheid om bestanden te muteren. Het exporteren van data naar een XML-bestand is niet een complexe functionaliteit.

CONCLUSIE

Omdat deze functionaliteit een essentieel gedeelte van de Extractor applicatie is en volledige toegang heeft tot het bestandssysteem van de computer waarop de applicatie wordt uitgevoerd heeft deze functionaliteit een zeer hoog risico.

1.2 VIEWER

1.2.1 XML-BESTAND IMPORTEREN

Het importeren van data vanuit een XML-bestand is een essentieel gedeelte van de Viewer applicatie. Deze functionaliteit heeft toegang tot het bestandssysteem van de computer waar de applicatie op uitgevoerd wordt, maar muteert geen data. Het importeren van data vanuit een XML-bestand is geen complexe functionaliteit.

CONCLUSIE

Doordat deze functionaliteit een essentieel gedeelte van de applicatie is, maar geen schrijftoegang heeft op het bestandssysteem heeft deze functionaliteit een hoog risico.

1.2.2 DATABASEDIAGRAM GENEREREN

Het genereren van een databasediagram en het tonen hiervan is de enige manier om data weer te geven in de Viewer applicatie. Deze functionaliteit wijzigt geen data, maar voert wel het primaire doel van de applicatie uit. Het genereren van een databasediagram heeft een relatief grote complexiteit.

CONCLUSIE

Deze functionaliteit muteert geen data, maar is wel van essentieel belang voor de applicatie. Het genereren en tonen van het databasediagram heeft een hoog risico.

1.2.3 EXPORTEREN NAAR XML-BESTAND

Het risico van het exporteren naar een XML-bestand is in de Viewer van minder belang dan in de Extractor applicatie. De functionaliteit van het exporteren en de rechten die hierbij horen zijn in de Viewer applicatie hetzelfde als in de Extractor applicatie. Het exporteren van data naar een XML-bestand is niet een complexe functionaliteit.

CONCLUSIE

Het exporteren van data naar een XML-bestand heeft in de Viewer een hoog risico.

1.2.4 INDELING DATABASEDIAGRAM WIJZIGEN

De functionaliteit waarbij de indeling van een databasediagram door de gebruiker gewijzigd kan worden is niet van essentieel belang in de Viewer applicatie, maar is wel vrij belangrijk in het overzichtelijk maken van de weergegeven data. Het wijzigen van de indeling van het databasediagram muteert data die in het geheugen is geladen, maar heeft geen effect op data buiten de applicatie. De indeling van het databasediagram wijzigen is relatief weinig complex.

CONCLUSIE

Het wijzigen van de indeling van een databasediagram heeft een middelmatig risico.

1.3 OVERZICHT

Onderstaande tabel toont de verschillende functionaliteiten van beide applicaties met bijbehorende bepaalde risiconiveau's.

Applicatie	Functionaliteit	Risiconiveau
Extractor	Databasestructuur inladen	Hoog
Extractor	Exporteren naar XML-bestand	Zeer hoog
Viewer	XML-bestand importeren	Hoog
Viewer	Databasediagram genereren	Hoog
Viewer	Exporteren naar XML-bestand	Hoog
Viewer	Indeling databasediagram wijzigen	Middelmatig

2. ALGEHELE CONCLUSIE

Zowel de Viewer als de Extractor applicatie bevatten vrijwel alleen functionaliteit met een hoog risiconiveau. Het testen van de functionaliteiten met een hoog tot zeer hoog risiconiveau vereist extra aandacht en moet in verschillende situaties getest worden.

Testscripts

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

Testscript_id	TG_1
Datum	26 februari 2010
Auteur	Ted de Koning
Korte omschrijving	Uitlezen database structuur
Testdoel	Het uitlezen van de structuur van een database
Testdatum	4 mei 2010
Tester	A. Hidalgo Aragones
Testresultaat (OK/NOK)	OK
Uitgebreide omschrijving	Het testen van de "Extractor", deze applicatie moet de structuur van een database uit lezen en wegschrijven naar een xml bestand.

Stap	Omschrijving van de actie	Verwachte Resultaat	Werkelijk Resultaat	Status (NOK, OK, Skip)
1.1	Applicatie opstarten	Applicatie is opgestart en het databaseconnectie scherm is getoond		OK
1.2	Klik op "Verbinden"	De applicatie toont melding "Niet alle velden zijn ingevuld."		OK
1.3	Onjuiste databasegegevens invoeren en op "Verbinden" klikken	Applicatie geeft aan dat er niet verbonden kon worden met de database		OK
1.4	Melding bevestigen door op "Ok" te klikken	Applicatie heeft het databaseconnectie formulier niet leeggehaald en toont het databaseconnectie scherm		OK
1.5	Juiste databasegegevens invoeren en op "Verbinden" klikken.	De applicatie toont een voortgangsbalk, wanneer het inladen van gegevens klaar is, wordt de melding "Gereed" weergegeven.		OK
1.6	Klik op "Ok"	De applicatie toont het XML exporteerscherm.		OK
1.7	Op "Bladeren" klikken	Bestandselectiescherm wordt getoond		OK
1.8	Niet bestaand bestandslocatie opgeven en op "Opslaan klikken"	Applicatie toont een melding dat de locatie onjuist is		OK
1.9	Melding bevestigen door op "Ok" te klikken	Applicatie heeft bestandslocatie leeggehaald en toont het XML bestandsselectie scherm		OK
1.10	Juiste bestandslocatie selecteren en op "Opslaan" klikken	De bestandslocatie is gekopieerd naar het tekstvak.		OK
1.11	Op "Exporteer" klikken	Applicatie toont voortgangsindicator, wanneer de applicatie klaar is met exporteren is wordt de melding "Klaar" weergegeven en er is een bestand aangemaakt en gevuld met gegevens uit de database		OK

1		**Testscript_id**	**Testgeval 1**		
2		**Datum**	2 maart 2010		
3		**Auteur**	Ted de Koning		
4		**Korte omschrijving**	Inlezen van een XML bestand waar een grafische representatie van opgebouwd moet worden		
5		**Testdoel**	Het tonen van een databasediagram aan de hand van een XML bestand		
6		**Testdatum**	4 mei 2010		
7		**Tester**	A. Hidalgo Aragones		
8		**Testresultaat (OK/NOK)**	OK		
9		**Uitgebreide omschrijving**	In dit testgeval wordt de importeer functionaliteit van de Viewer applicatie getest.		
10					
11	**Stap**	**Omschrijving van de actie**	**Verwachte Resultaat**	**Werkelijk Resultaat**	**Status (NOK, OK, Skip)**
12	**1.1**	**Start de applicatie op**	De applicatie is opgestart en toont het importeerscherm		OK
13	**1.2**	**Klik op "Importeer"**	De applicatie toont de foutmelding "Geen bestand geselecteerd."		OK
14	**1.3**	**Klik op "Bladeren"**	De applicatie toont een bestandsselectiescherm.		OK
15	**1.4**	**Selecteer bestand zonder XML data en klik op "Openen"**	De applicatie heeft de locatie en de naam van het bestand in het tekstvak geplaatst.		OK
16	**1.5**	**Klik op "Importeer"**	De applicatie toont de foutmelding "Het XML-document (1, 1) bevat een fout."		OK
17	**1.6**	**Klik op "Bladeren"**	De applicatie toont een bestandsselectiescherm.		OK
18	**1.7**	**Klik op "Importeer"**	De applicatie toont een voortgangsindicator en importeert het bestand. Wanneer de applicatie klaar is toont de applicatie de melding "Klaar met importeren"		OK

Testscript_id	Testgeval 2	
Datum	2 maart 2010	
Auteur	Ted de Koning	
Korte omschrijving	De locatie van een tabel in het databasediagram wordt gewijzigd, deze wijziging wordt opgeslagen in een XML bestand	
Testdoel	Het opslaan van locaties in XML document	
Testdatum	4 mei 2010	
Tester	A. Hidalgo Aragones	
Testresultaat (OK/NOK)	OK	
Uitgebreide omschrijving	In dit testgeval wordt de functionaliteit getest van de Viewer applicatie. De te testen functionaliteit omvat het importeren van een XML-bestand, genereren van een diagram aan de hand van de geïmporteerde data en het opslaan van een gewijzigde versie van deze data.	

Stap	Omschrijving van de actie	Verwachte Resultaat	Werkelijk Resultaat	Status (NOK, OK, Skip)
1.1	Start de applicatie op	De applicatie toont het XML importeerscherm		OK
1.2	Klik op "Bladeren"	De applicatie toont het bestandsselectiescherm		OK
1.3	Klik op "Importeer" en selecteer XML bestand	De applicatie toont een voortgangsindicator, wanneer de applicatie klaar is met importeren wordt de melding "Klaar met importeren" getoond.		OK
1.4	Klik op "Toon diagram"	De applicatie toont een blauw scherm met een balk aan de rechterkant met de tekst "Diagram elementen"		OK
1.5	Klik op "Toon volledig diagram"	De applicatie toont een diagram met de elementen die aanwezig zijn in het XML bestand.		OK
1,6	Sleep een tabel van links naar rechts	De tabel is verplaatst naar de gewenste locatie en de relatielijnen zijn aangepast aan deze nieuwe locatie		OK
1.7	Klik op "Bestandstoegang" in de linker navigatiebalk	De applicatie toont een lijst met importeer en exporteer opties.		OK
1,8	Klik op "Exporteer naar XML"	De applicatie toont het XML exporteerscherm		OK
1.9	Klik op "Bladeren"	De applicatie toont het bestandsselectiescherm		OK
1.10	Selecteer een bestandslocatie waar schrijfrechten beschikbaar voor zijn en klik op "Opslaan"	De applicatie exporteert het databasediagram naar de opgegeven bestandslocatie		OK
1.11	Klik op "Importeer van XML"	De applicatie toont het XML importeerscherm		OK
1.12	Klik op "Bladeren"	De applicatie toont een bestandsselectie scherm		OK
1.13	Geeft de bestandslocatie op waar zojuist naar geëxporteerd is en klik op "Openen"	De applicatie toont een voortgangsindicator, wanneer de applicatie klaar is met importeren wordt de melding "Klaar met importeren" getoond.		OK
1.14	Klik op "Toon volledig diagram" in de linker navigatiebalk	De applicatie toont een diagram met de elementen die aanwezig zijn in het XML bestand en het verplaatste element.		OK

Testrapport

Ontwikkelen database diagram generator voor Centric IT Solutions

Auteur	T.F. de Koning
Studentnummer	20062351
Examinatoren	Mvr. R. Bechet Dhr. M. Reijnhoudt
Bedrijfsmentor	Dhr. P.J.H.T.P.M. van Rijswijk
Projectnaam	Ontwikkelen database diagram generator voor Centric IT Solutions
Studie	Informatica
Afstudeerperiode	08-02-2010 t/m 04-06-2010

INHOUDSOPGAVE

INLEIDING

Dit document beschrijft de bevindingen en resultaten van de tests die uitgevoerd zijn op de Extractor en Viewer applicaties ontwikkeld binnen het afstudeerproject van Ted de Koning.
De Extractor en Viewer applicaties behoren tot één systeem waarmee het mogelijk moet zijn om de structuur van een Oracle database uit te lezen en als databasediagram op het scherm te tonen.

Het eerste hoofdstuk geeft een korte indruk van de resultaten en geeft informatie over de status van het geteste systeem. Het tweede hoofdstuk bevat de resultaten van de schermtests. Hoofdstuk 3 tot en met 6 beschrijft de status van het geteste systeem. Het laatste hoofdstuk toont de resultaten van de performancetests.

1. CONCLUSIE

Uit de uitgevoerde tests is gebleken dat de applicaties niet gereed zijn voor productiegebruik. Er wordt aangeraden om de openstaande bevindingen op te lossen voordat de applicaties in gebruik worden genomen.

Resultaten:

- De testdekking is compleet
- Er staan in totaal **Negen** bevindingen open waarvan één met erg hoge prioriteit.

1.1 CONCLUSIE, PER TESTOBJECT

1.1.1 EXTRACTOR

De Extractor applicatie heeft in totaal drie bevindingen open staan, waarvan er geen een hogere prioriteit heeft dan midden.
De performance in de Extractor is volgens de specificaties in het testplan en de testscripts zijn succesvol uitgevoerd.

1.1.2 VIEWER

De Viewer applicatie heeft in totaal zes bevindingen open staan, waarvan één met de prioriteit hoog.
De performance in de Viewer is volgens de specificaties in het testplan, maar er kan geen volledig diagram van de TestMV database opgebouwd worden. De testscripts zijn wel succesvol uitgevoerd.

2. SYSTEEMTESTS

2.1 SCHERMTESTS

2.1.1 EXTRACTOR

DATABASECONNECTIESCHERM

Op het moment dat de Extractor opgestart is, wordt het Databaseconnectiescherm getoond. Het scherm laadt zonder foutmeldingen.
Wanneer er via de navigatiebalk naar het databaseconnectiescherm genavigeerd wordt, laadt het scherm ook zonder foutmeldingen, zijn de tekstvelden leeggehaald en is de voortgangsindicator niet zichtbaar.

XML-EXPORT SCHERM

Het XML-export scherm wordt geladen nadat de gebruiker een database succesvol geëxporteerd heeft. Het XML-export scherm laad tevens zonder foutmeldingen. Wanneer er via de navigatiebalk naar het XML-export scherm genavigeerd wordt, laad het scherm zonder foutmeldingen, is het tekstvak leegehaald en de voortgangsindicator is, net als bij het databaseconnectiescherm, verdwenen.

2.1.2 VIEWER

XML-IMPORTEERSCHERM

Het XML-import scherm wordt getoond zodra de applicatie opgestart is, het scherm laad zonder foutmeldingen. Wanneer er via de navigatiebalk naar het XML-importeerscherm genavigeerd wordt, laad het scherm ook zonder foutmeldingen, is het tekstvak leeggehaald en is de voortgangsindicator niet zichtbaar.

VOLLEDIG DIAGRAM SCHERM

Het volledig diagram scherm wordt getoond zodra er geklikt wordt op "Toon volledig diagram" in de navigatiebalk. Wanneer de gebruiker op "Toon volledig diagram" klikt, wordt er een voortgangsindicator getoond en wordt het diagram opgebouwd op het scherm. Wanneer er geen data in het geheugen is ingeladen wordt er een melding weergegeven.
Dit scherm werkt zonder fouten.

3. STATUS VAN TESTOBJECTEN

De status van de testobjecten wordt bepaald door het aantal bevindingen. Naast het aantal is ook de prioriteit van de bevindingen van belang voor de status.
Er zijn drie verschillende statussen gedefinieerd:

- Compleet (Icoon )
- Compleet, niet werkend (Icoon )
- Niet werkend (Icoon )

Wanneer een testobject nul bevindingen open heeft staan, wordt de status "Compleet" toegekend. De status "Compleet, niet werkend" wordt toegekend wanneer het testobject bevindingen open heeft staan, maar geen van de bevindingen heeft een hoge prioriteit. De status "Niet werkend" wordt toegekend aan een testobject met minimaal één bevinding met een hoge prioriteit.

Testobjecten	Status	Openstaande bevindingen
Extractor		3
Viewer		6

4. BEVINDINGEN

4.1 OPENSTAANDE BEVINDINGEN, PER TESTOBJECT

Onderstaande tabel geeft aan hoeveel bevindingen er per testobject open staan en welke prioriteit ze hebben.

Testdoel	Erg hoog	Hoog	Midden	Laag	Totaal
Extractor	0	0	1	2	3
Viewer	1	0	2	3	6
Totaal	**1**	**0**	**3**	**5**	**9**

Uitleg prioriteit criteria:

Erg hoog
De bevinding blokkeert de volledige werking van het testobject of een groot gedeelte ervan.

Hoog
De bevinding blokkeert een klein gedeelte van het testobject.

Midden
De bevinding stopt niet de werking van het testobject, maar is wel storend voor de gebruiker.

Laag
De bevinding omvat een fout zonder enige impact op de werking van het testobject.

4.2 OPENSTAANDE BEVINDINGEN

Onderstaande tabel toont de bevindingen opgedaan tijdens het uitvoeren van de testscripts en de systeemtests. De bevindingen zijn aangeduid met een prioriteit niveau waarmee de ernst van de bevinding aangeduid wordt.

Bevinding	Omschrijving	Applicatie	Prioriteit
1	Voortgangsbalk ophalen databasestructuur werkt niet goed.	Extractor	Midden
2	Tonen van groot diagram in Viewer duurt lang.	Viewer	Midden
3	Slepen met de muis op de naam van diagramelement werkt niet.	Viewer	Laag
4	Grote diagramelementen worden soms over elkaar geplaatst.	Viewer	Midden
5	Tonen van complete TestMV testdatabase is onmogelijk.	Viewer	Erg hoog
6	Inklappen van navigatiemenu is niet mogelijk via een knop.	Beide	Laag
7	Foutmeldingen worden soms niet volledig weergegeven.	Beide	Laag

5. STATUS TESTSCRIPTS

Tijdens het uitvoeren van de verschillende systeem- en acceptatietests zijn er testscripts uitgevoerd. Testscripts zijn tests waarbij in stappen beschreven wordt wat de tester uit moet voeren om te bevestigen of de testobjecten werken naar behoren.

Testdoel	Automatisch	Testdocumentatie	Testdekking %
Extractor	N	J	100%
Viewer	N	J	100%

6. RESULTATEN PERFORMANCETEST

Alle tests zijn uitgevoerd op één computer waarbij de databaseserver op het interne netwerk aanwezig was. De gebruikte database is de testmv database van Centric, deze database wordt vaker gebruikt voor het ontwikkelen en testen van applicaties bij Centric.

6.1 EXTRACTOR

6.1.1 IMPORTEREN VAN DATABASESTRUCTUUR

Het importeren van de TestMV database wordt gedaan in de Extractor applicatie. Het proces haalt de benodigde gegevens op uit de database en laad deze in het geheugen van de applicatie. De gebruiker moet de databaseserver, de databasenaam, de gebruikersnaam en het wachtwoord opgeven om met de database te kunnen verbinden. De test is vier keer uitgevoerd in dezelfde situatie als eerder beschreven is.

Testrun	Tijdsduur van importeren testmv database
1	150 seconden
2	149 seconden
3	152 seconden
4	150 seconden
Gemiddeld	**150,25 seconden**

Uit bovenstaande tabel is gebleken dat het importeren van de TestMV database gemiddeld 152,25 seconden duurt. De tijdsduur valt binnen het vastgestelde limiet van vijf minuten (300 seconden). De applicatie is geslaagd voor deze performance test.

6.1.2 EXPORTEREN NAAR XML-BESTAND

Als uitbereiding op voorgaande performance test, wordt de ingeladen data weggeschreven naar een XML-bestand. Het XML-bestand wordt gebruikt om de data op te slaan en kan ingeladen worden in de Viewer applicatie.

Testrun	Tijdsduur van exporteren testmv database naar XML-bestand
1	5 seconden
2	4 seconden
3	4 seconden
4	5 seconden
Gemiddeld	**4,5 seconden**

Bovenstaande tabel toont de testresultaten van het exporteren van de ingeladen data naar een XML-bestand. Uit de resultaten blijkt dat het exporteren gemiddeld 4,5 seconden duurt en valt hiermee ruim binnen de grens van 30 seconden. De applicatie is geslaagd voor deze performance test.

6.2 VIEWER

6.2.1 IMPORTEREN VAN XML-BESTAND

In deze test wordt het XML-bestand dat tijdens het testen van de Extractor geëxporteerd is, geïmporteerd in de Viewer. Het importeerproces zorgt ervoor dat de data die in het XML-bestand staat in het geheugen van de applicatie geladen wordt, zodat de applicatie een diagram van de data kan genereren.

Testrun	Tijdsduur importeren van XML-bestand
1	3 seconden
2	4 seconden
3	3 seconden
4	3 seconden
Gemiddeld	**3,25 seconden**

Bovenstaande tabel toont de testresultaten van het importeren van een XML-bestand met de data uit de TestMV database. De gemiddelde tijdsduur van dit proces is 3,25 seconden en valt hiermee ruim binnen de grens van 30 seconden. De applicatie is geslaagd voor deze performance test.

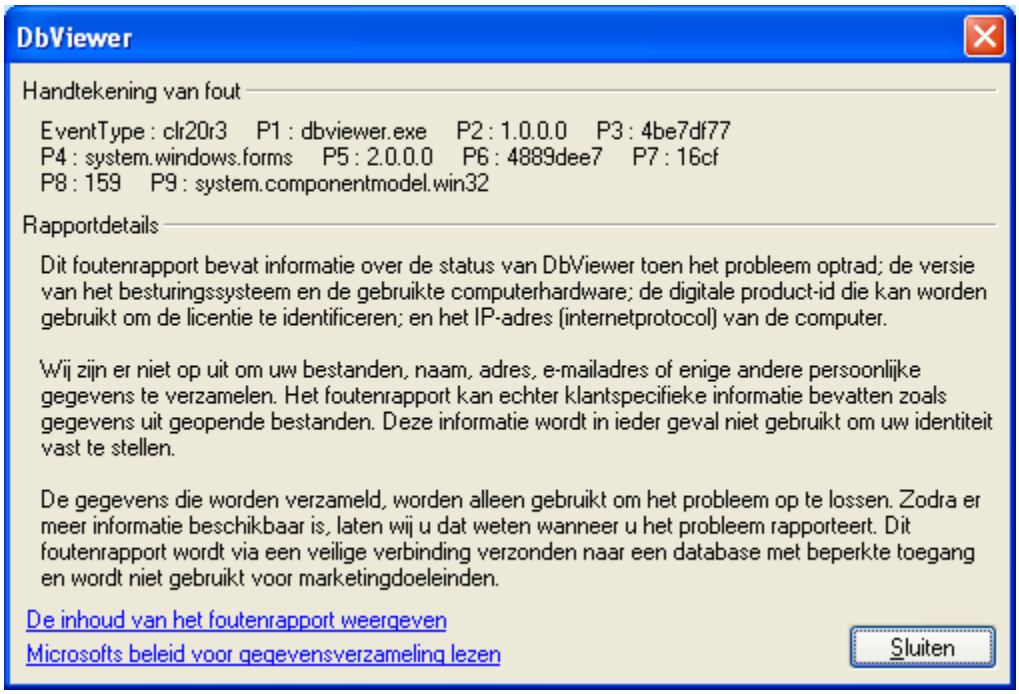
6.2.2 GENEREREN VAN DATABASEDIAGRAM

Bij het genereren van het databasediagram wordt het proces, waarbij de elementen van het databasediagram op het scherm gezet worden, bedoeld. Voor deze test wordt het XML-bestand gebruikt dat door de Extractor applicatie gegenereerd is.

Testrun	Tijdsduur van genereren databasediagram
1	Error
2	Error
3	Error
4	Error
Gemiddeld	**geen**

De applicatie toont een foutmelding zodra alle elementen op het scherm gezet zijn. Het probleem komt niet voor wanneer er kleinere databases ingeladen worden. De applicatie is voor deze performance test niet geslaagd.

Details van de foutmelding worden in afbeelding 1 weergegeven.



Afbeelding 1, Foutmelding van foutmelding in Viewer bij teveel data.

Doordat de complete TestMV database niet getoond kan worden, wordt er een aangepaste variant gebruikt in de volgende performance test. Deze aangepaste variant bevat, met uitzondering van twee schema's, alle schema's van de TestMV database. Door deze aanpassing is het wel mogelijk om met een grote datacollectie te werken in de Viewer en ontstaat er geen foutmelding.

Testrun	Tijdsduur genereren databasediagram aangepaste TestMV
1	179 seconden
2	181 seconden
3	176 seconden
4	182 seconden
Gemiddeld	**179,5 seconden**

Het gemiddelde resultaat van 179,5 seconden valt net binnen het acceptabele limiet van 180 seconden. De snelheid van het genereren van het databasediagram is afhankelijk van de snelheid van de computer waarop de applicatie uitgevoerd wordt. De applicatie is voor deze performancetest geslaagd.

6.2.3 SLEPEN VAN TABELLEN

Aangezien het tonen van het complete databasediagram niet werkt met de complete TestMV database, zijn er twee databaseschema's uit het XML-bestand gehaald om er toch voor te zorgen dat er getest kan worden met een grote database met ongeveer 1800 elementen.

Het slepen van de elementen in het aangepaste TestMV databasediagram wordt zonder wachttijden uitgevoerd. De gebruiker heeft geen last van onderbrekingen tijdens het werken met het complete diagram en de relaties worden na het slepen meteen getoond met de nieuwe coördinaten. De applicatie is voor deze performancetest geslaagd.

7. STATUS TESTEN VAN ONDERDELEN

Dit hoofdstuk toont een korte indruk van de status van de testobjecten, zodat in één oogopslag de belangrijkste bevindingen waargenomen kunnen worden.

Testdoel	Opmerkingen
Extractor	Meldingen worden niet volledig getoond, voortgangsbalk werkt niet naar behoren.
Viewer	Meldingen worden niet volledig getoond, performance mag beter, testdatabase kan niet volledig worden getoond.

Bespreking concept	**Tussentijds assessment** Ted de Koning	Eerste beoordeling

[1]**betreft:**	Ted de Koning (20062351)
geplande eerstvolgend TTTA:	dinsdag 11 mei 2010
realisatie eerste TTA:	woensdag 19 mei 2010
realisatie (evt) tweede TTA:	(in te vullen door de coordinator)

	Tijdens het tussentijds assessment is het volgende geconstateerd:	**ja:**
a	Het voortgangsverslag is ontvangen	☒
b	Het afstudeerdossier is beschikbaar op Blackboard	☒
c	Het afstudeerdossier is opgebouwd conform de richtlijnen	☒
d	Het goedgekeurde afstudeerplan is aanwezig	☒
e	Het plan van aanpak is aanwezig	☒
f	Reeds geleverd commentaar is aanwezig	☐
g	Het afstudeerdossier geeft voldoende inzicht in de stand van zaken	☒
h	De afstudeeropdracht is tot nu toe naar behoren uitgevoerd	☒

Overige beoordelingscriteria		**V**	**toelichting indien niet voldoende:**
Aanpak			
	Passend	☒	
	Theoretisch verantwoord	☐	**de gehanteerde testmethode moet beter worden beschreven**
Producten			
	Tussenproducten	☐	
	Eindproduct	☐	

[1] **Dit formulier wordt door de expert-examinator digitaal ingevuld, waarna de begeleider-examinator het per email verstuurt naar de student met een cc naar de coördinator van ICT & Media @ Work. Het formulier dient door de student te worden opgenomen in het afstudeerdossier.**

m.reijnhoudt@hhs.nl**;**
H.G.J.Bechet-Tjoonk@hhs.nl**;**
20062351@student.hhs.nl;
a.m.schipper@hhs.nl ;
i-afstuderen@hhs.nl

Bespreking concept	**Tussentijds assessment** Ted de Koning	Eerste beoordeling

Reflectie			
	Inzicht in eigen functioneren	☒	
	Inzicht in eigen leerproces	☒	

Beroepstaken op afgesproken niveau uitgevoerd?

code beroepstaak	nummer activiteit	niveau	(evt) tekst van de beroepstaak / activiteit	het behaald zijn van deze beroepstaak wordt aangetoond door: • een produkt uit het afstudeerdossier, genoemd in punt 9 van afstudeerplan, deel #1, of • het eindverslag	wordt opgeleverd in het afstudeerdossier in weeknr …, conform het afstudeerplan, deel #4: de activiteitenplanning
1.4		**4**	**Uitvoeren analyse door definitie van requirements**	- Use cases - Domeinmodel - Inception rapport	**3** **4** **3**
3.1		**3**	**Ontwerpen softwarearchitectuur**	- Plan van aanpak - Use-case diagrammen - Deployment diagram - Klassediagram - Sequence diagrammen	**2** **3** **5** **5** **6**
3.3		**3**	**Bouwen applicatie**	Construction rapport	**15**
3.4		**3**	**Initiëren en plannen van het testproces**	- Testplan	**10**
3.5		**3**	**uitvoeren van en rapporteren over het testproces**	- Testrapport	**15**

Toelichting:

Bespreking concept	**Tussentijds assessment** Ted de Koning	Eerste beoordeling

Advies: Besteed de komende tijd uitsluitend aan de verslaggeving en een betere beschrijving van de opgeleverde produkten

Aankruisen welke beslissing de student heeft genomen:

☒	**Afstudeerdossier wordt, zoals afgesproken in het afstudeerplan, ingeleverd op** vrijdag 4 juni 2010
☐	**De afstudeerperiode wordt met 30% verlengd,** **nieuwe einddatum** donderdag 8 juli 2010 **nieuwe inleverdatum:**
☐	**Student stopt met de afstudeeropdracht**

begeleider - examinator: H.G.J. Bechet-Tjoonk

expert - examinator: M. Reijnhoudt

Datum:

Bespreking concept	**Tussentijds assessment** Jimmy Rijkeboer	Eerste beoordeling

Bespreking concept	Tussentijds assessment	Eerste beoordeling

Formulier bespreking concept afstudeerdossier

Student: Ted de Koning **Studentnummer:** 20062351

Datum: 15-4-2010

Tijdens de bespreking is het volgende geconstateerd:		**ja**	**nee**
a	Het voortgangsverslag is ontvangen	**x**	
b	Het afstudeerdossier is beschikbaar op Blackboard	**x**	
c	Het afstudeerdossier is opgebouwd conform de richtlijnen		**x**
d	Het goedgekeurde afstudeerplan is aanwezig		**x**
e	Het plan van aanpak is aanwezig	**x**	
f	Reeds geleverd commentaar is aanwezig nvt		
g	Het afstudeerdossier geeft voldoende inzicht in de stand van zaken	**x**	
h	De afstudeeropdracht is tot nu toe naar behoren uitgevoerd	**x**	

Verbeterpunten:

Eindverslag een goede structuur geven en veel schrijven!

Opmerkingen:

Naam begeleider/examinator: **Rianne Bechet**

Datum:15-04-2010

Dit formulier wordt door de begeleider/examinator digitaal ingevuld en per email naar de student verstuurd met een cc naar de coördinator van ICT & Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2010-1.1
Startdatum: 8 februari 2010
Einddatum: 4 juni 2010
Inleverdatum: 4 juni 2010

Studentnummer: 20062351
Achternaam: Koning
Voorletters: T.F.
Roepnaam: Ted
Adres: Geerweg 2
Postcode: 2631PD
Woonplaats: Nootdorp
Telefoon: 06 452 556 30

Opleiding: INF
Locatie: Den Haag
Variant: voltijd

Naam studieloopbaanbegeleider: B.M. Derks
Naam begeleider/examinator: H.G.J. Bechet-Tjoonk
Naam expert/examinator: M. Reijnhoudt

Naam bedrijf: Centric
Afdeling bedrijf:
Bezoekadres bedrijf: Antwerpseweg 8
Postcode bezoekadres: 2803PB
Postbusnummer: 338
Postcode postbusnummer:
Plaats: Gouda
Telefoon bedrijf: +31 182 34 50 00
Telefax bedrijf:
Internetsite bedrijf: www.centric.nl

Achternaam opdrachtgever: mw / dhr van Rijswijk
Voorletters opdrachtgever: *P.J.H.T.P.M*
Titulatuur opdrachtgever:
Functie opdrachtgever: manager
Doorkiesnummer opdrachtgever: 0182-345204
Email opdrachtgever: Patrick.van.Rijswijk@centric.eu

Achternaam bedrijfsmentor:
Voorletters bedrijfsmentor:
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor:
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor:

Doorkiesnummer afstudeerder:
Bedrijfsemail afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht: Ontwikkelen database diagram generator

Afstudeerplan, deel #1: Opdrachtomschrijving

1. Bedrijf

Centric IT Solutions is een bedrijf dat aan veel verschillende soorten bedrijven en overheidsinstanties oplossingen door middel van software aanbiedt.
Veel van de software die Centric levert, is te koppelen met andere applicaties van Centric.
Naast het leveren van standaard software pakketten, ontwikkelt Centric ook maatwerk indien er specifieke wensen zijn van de klant.
Het ontwikkelen van applicaties en maatwerk wordt door medewerkers van Centric zelf gedaan. Centric maakt geen gebruik van outsourcing omdat de eigenaar van Centric de kennis intern wenst te houden.
De afdeling waar de afstudeeropdracht wordt uitgevoerd heeft vooral lokale overheden als klanten. De applicaties die op deze afdeling ontwikkeld worden zijn gericht op bouw- en woningtoezicht, bouwvergunningen en milieuvergunningen. Op de afdeling worden applicaties ontworpen, gebouwd en getest, tevens worden systeembeheerders begeleid om systemen in te richten of updates uit te voeren. Daarnaast worden ook trainingen gegeven aan eindgebruikers van de software.
Ook is er op de afdeling een helpdesk aanwezig, waar eerste- en tweedelijns ondersteuning aan eindgebruikers geleverd wordt.

2. Aanleiding

Vanuit Centric is er vraag naar een manier waarmee databasestructuren eenvoudig weergegeven kunnen worden. De gegenereerde overzichten zullen gebruikt worden om maatwerk te vereenvoudigen en om systeembeheerders bij klanten te ondersteunen.

3. Probleemstelling

Op dit moment kunnen medewerkers van Centric alleen de structuur van databases bekijken door middel van de interface van de Oracle DBMS en is er geen manier om relaties tussen verschillende databases in te zien. Tevens wordt er regelmatig van de klant verzocht om een databasediagram van de applicaties te versturen, dit is op dit moment niet mogelijk.

4. Doelstelling van de afstudeeropdracht

Het hebben van een applicatie die inzicht biedt in relaties van verschillende databases, die tevens met andere databases relaties kunnen hebben.
De op te leveren applicatie dient generiek voor Oracle databases toepasbaar te zijn.

5. Resultaten voor de opdrachtgever

Tijdens de afstudeerperiode wordt een werkende applicatie opgeleverd waarmee databasediagrammen gegenereerd kunnen worden aan de hand van bestaande Oracle databases.

De opdrachtgever ontvangt, tijdens de afstudeerperiode de volgende producten:

- Plan van aanpak
- Werkende applicatie waarmee databases in databasediagrammen weergegeven kunnen worden inclusief source code.
- Unit-tests
- Use-case diagrammen
- Requirements
- Testplan
- Klassediagrammen
- Sequence diagrammen

- Deployment diagram
- Testrapport
- Domeinmodel
- Use-case diagrammen

6. Uitgangssituatie aan de hand van:

a. Beschikbare noodzakelijke software (*indien van toepassing*)

- Visual Studio
- Oracle Database
- SVN, of andere versiebeheer software.

b. Beschikbare noodzakelijke hardware (*indien van toepassing*)

N.v.t.

c. Al opgeleverde relevante documenten

Geen.

d. Aanwezige ideeën

De opdrachtgever eist dat de applicatie uit meerdere onderdelen bestaat, waardoor toekomstig onderhoud eenvoudig wordt. Hiernaast opdrachtgever eist dat de applicatie object georiënteerd ontwikkeld wordt. Tevens wordt er verwacht dat er unit-tests bij de ontwikkelde classes komen. Tevens moet er rekening gehouden worden met toekomstige functionaliteit, zoals het wijzigen van databasediagrammen in de applicatie.

Het moet mogelijk zijn om de gegenereerde diagrammen te kunnen exporteren naar een XML bestand. De geëxporteerde XML bestanden moeten geïmporteerd kunnen worden in de applicatie zodat technische medewerkers van klanten deze ook in kunnen zien.

7. Werkzaamheden aan de hand van:

a. Te hanteren methodieken

Centric wenst dat het ontwikkelproces volgens een Agile ontwikkel methode gebeurt. Bij Centric is er weinig ervaring met Agile ontwikkel methodes, maar er wordt onderzoek gedaan welke methode er het beste bij Centric past.

Als ontwikkelmethode is er gekozen voor Agile Unified Process. Er is voor Agile Unified Process (AUP) gekozen omdat dit een combinatie is tussen een Agile ontwikkelmethode en, het veel traditionelere, Rational Unified Process (RUP). Hierdoor kan Centric kennis maken met een ontwikkelmethode die zowel traditionele software ontwikkeling als Agile combineert.

Tijdens het ontwikkelen van de applicatie zal er gewerkt worden volgens Test Driven Development. Test Driven Development is een techniek die binnen AUP wordt aangeraden om de productiviteit en kwaliteit te verhogen.

b. Uit te voeren activiteiten
Er wordt een plan van aanpak opgesteld, waarin onder andere de afbakening en planning gedefinieerd zal worden.

Tijdens de inception fase worden de requirements van de applicatie vastgelegd. Deze requirements zullen de rest van het ontwerp bepalen.
Naast de requirements worden er ook use-case diagrammen, met bijbehorende beschrijvingen, gemaakt. In deze use-cases worden de acties die mogelijk zijn binnen de applicatie vastgelegd.

Binnen de elaboration fase wordt de nadruk op het ontwerpen van de architectuur van de applicatie gelegd. Er zal een klassediagram worden gemaakt waardoor de structuur van de applicatie vastgelegd wordt. Tevens worden er sequence diagrammen gemaakt, zodat de interactie tussen de verschillende componenten wordt gedefinieerd.

Er wordt er een testplan opgesteld. Het testplan bevat onder andere een testrisicoanalyse, testdoelen en de soorten tests die uitgevoerd moeten worden.

Tijdens de Construction fase zal het implementeren van het ontwerp, dat is ontstaan tijdens de inception- en elaboration fase, gerealiseerd worden. Tijdens het implementeren van het ontwerp zal gebruik worden gemaakt van Test Driven Development, door middel van unit-tests. Tevens wordt in deze fase het testplan uitgevoerd en worden er PSD's gemaakt.

c. Te gebruiken technieken

De applicatie wordt ontwikkeld in .NET, specifiek in C#, en zal een desktop applicatie worden. Tijdens het ontwikkelen van deze applicatie zal de techniek Test Driven Development gebruikt worden.

d. Te vermelden nadrukken

De nadruk van de te ontwikkelen applicatie, zal liggen op het genereren van databasediagrammen aan de hand van bestaande databases, zodat inzicht in de relaties van deze databases mogelijk is.

8. Risico's en maatregelen

- Risico: Geen medewerking van Centric.
  Maatregel: Situatie bespreken met begeleider van de HHS.
- Risico: Gebrek aan kennis bij ontwikkelen van grafische weergave van tabelstructuren in diagramvorm.
  Maatregel: Bespreken met opdrachtgever, eventueel een expert bij Centric raadplegen.
- Risico: Er zijn geen databases beschikbaar om de ontwikkeling van de applicatie te ondersteunen.
  Maatregel: Bespreken met opdrachtgever, eventueel eigen testdatabases maken.

9. (Tussen)producten die opgenomen worden in het afstudeerdossier

Voor een deel zijn dit waarschijnlijk produkten die voortvloeien uit de beschrijving van punt 5: resultaten voor de opdrachtgever, maar hier moeten ook de produkten worden genoemd die de student in zijn afstudeerdossier gaat opnemen, omdat ze genoemd worden worden afstudeerplan, deel #3.

- Plan van aanpak
- Eindverslag
- Inception rapport
  - Use-case diagrammen
  - Requirements

- Testplan
- Elaboration rapport
  - Domeinmodel
  - Klassediagrammen
  - Sequence diagrammen
  - Deployment diagram
- Construction rapport
  - Testrapport
  - PSD's

o Afstudeerplan, deel #2: Benodigde Beroepstaken / activiteiten

De code van de beroepstaak loopt van 1.1 tot en met AV-4
Het nummer van de activiteiten loopt van 1 tot en met 57

code beroepstaak	nr activiteit	niveau	behaald in majorblok	(evt) tekst van de competentie
2.2		2	I-5	
3.1		3	I-2, I-6	
3.2		3	I-2, I-6	
3.3		3	I-2, I-6	
4.1		2	I-4	

Afstudeerplan, deel #3: Te demonstreren beroepstaken en wijze waarop

code beroepstaak	nummer activiteit	niveau	(evt) tekst van de beroepstaak / activiteit	het behaald zijn van deze beroepstaak wordt aangetoond door: • een produkt uit het afstudeerdossier, genoemd in punt 9 van afstudeerplan, deel #1, of • het eindverslag	wordt opgeleverd in het afstudeerdossier in weeknr …, conform het afstudeerplan, deel #4: de activiteitenplanning
1.4		4	Uitvoeren analyse door definitie van requirements	- Use cases - Domeinmodel - Inception rapport	3 4 3
3.1		3	Ontwerpen softwarearchitectuur	- Plan van aanpak - Use-case diagrammen - Deployment diagram - Klassediagram - Sequence diagrammen	2 3 5 5 6
3.3		3	Bouwen applicatie	Construction rapport	15
3.4		3	Initiëren en plannen van het testproces	- Testplan	10
3.5		3	uitvoeren van en rapporteren over het testproces	- Testrapport	15

Persoonlijke verantwoording van keuze opdracht/bedrijf/Beroepstaken

De reden om bij Centric af te studeren was omdat ik al een aantal jaar bij een klein ICT bedrijf werk. Ik wil ervaren hoe er gewerkt wordt in een groot bedrijf. Tevens zocht ik een afstudeerplek waarbij er met .NET gewerkt wordt. .NET wordt niet erg uitgebreid behandeld in de opleiding, maar wordt erg veel gevraagd.

Ik heb gekozen om een desktop applicatie in .NET te ontwikkelen omdat ik al veel ervaring heb met het ontwikkelen van webapplicaties in, onder andere, PHP. Met deze afstudeeropdracht wil ik mijn kennis verbreden, zodat ik een betere plek in de arbeidsmarkt krijg.

Nominale duur afstudeerperiode 0 weken.

Afstudeerplan, deel #4: Activiteitenplanning

uit te voeren activiteiten / gebruikte technieken	naam van het resulterend produkt	weeknr …. beginnend bij 1 en eindigend minimaal bij 17
- Afbakening van de opdracht - Opstellen planning	Plan van aanpak	02
- Maken use-cases - Opstellen van requirements	Inception rapport	03
- Ontwerpen sequence diagram - Ontwerpen klassediagram - Ontwerpen deployment diagram	Elaboration rapport	05
- Opstellen testdoelen - Maken van testrisicoanalyse - Vaststellen van testobjecten	Testplan	10
- Bouwen van applicatie - Maken van Unit-tests - Maken van PSD’s - Uitvoeren testplan	Construction rapport	15

	datum:	handtekening:
student:		
opdrachtgever-bedrijfsmentor P.J.H.T.P.M van Rijswijk		
begeleider-examinator H.G.J. Bechet-Tjoonk	**woensdag, februari 02, 2010**	
expert-examinator M. Reijnhoudt	**(nog onbekend)**	