## 

|  |
| --- |
| Fontys |
| Ontwerpdocument |
| Ontwerpdocument |

|  |
| --- |
| Sander Koch S21M  20-1-2016 |

## 

Inhoud

[Inleiding 3](#_Toc441066121)

[ERD 4](#_Toc441066122)

[Relationeel database ontwerp 5](#_Toc441066123)

[Datamodel 5](#_Toc441066124)

[Klassendiagram 7](#_Toc441066125)

[Domeinmodel 7](#_Toc441066126)

[Implementatie model 8](#_Toc441066127)

## Inleiding

In dit document kunt u de ontwerpfase van ‘t Sloepske applicatie vinden. U kunt informatie vinden over de database ontwerp Vervolgens kunt u ook informatie vinden over de klassenstructuur, door middel van de Klassendiagram en toelichting daarvan.

## Database ontwerp

In tegenstelling tot de ERD, heeft de database ontwerp wel alle attributen/entiteiten uitwerkt. Hierbij kunt u al zien hoe de uit eindelijke database er uit gaat zien. Met de foreign key relaties aan geduid.

In plaats van de datamodel in de strokendiagram te zetten, heb ik er voor gekozen om dit in aparte tabellen te doen. Deze vind u hier onder, en op de volgende pagina. Hierbij zijn alle datatypes van elke attribuut weergegeven, en of ze verplicht ingevuld moeten zijn.

### Datamodel

|  |  |  |
| --- | --- | --- |
| **Gebied** | | |
| **Kolom** | **Datatype** | **Verplicht?** |
| ID | NUMBER | Ja |
| Naam | VARCHAR2 | Ja |
| Kaart | VARCHAR2 | Ja |

|  |  |  |
| --- | --- | --- |
| **Project** | | |
| **Kolom** | **Datatype** | **Verplicht?** |
| ID | NUMBER | Ja |
| Naam | VARCHAR2 | Ja |
| Gebied\_ID | NUMBER | Ja |

|  |  |  |
| --- | --- | --- |
| **Bezoek** | | |
| **Kolom** | **Datatype** | **Verplicht?** |
| ID | NUMBER | Ja |
| Project\_ID | VARCHAR2 | Ja |
| BeginTijd | DATE | Ja |
| EindTijd | DATE | Nee |

|  |  |  |
| --- | --- | --- |
| **Waarneming** | | |
| **Kolom** | **Datatype** | **Verplicht?** |
| ID | NUMBER | Ja |
| Bezoek\_ID | NUMBER | Ja |
| Soort\_Dier | NUMBER | Ja |
| X | NUMBER | Ja |
| Y | NUMBER | Ja |
| Type\_Waarneming | NUMBER | Ja |

|  |  |  |
| --- | --- | --- |
| **Diersoort** | | |
| **Kolom** | **Datatype** | **Verplicht?** |
| ID | NUMBER | Ja |
| Type | VARCHAR2 | Ja |
| Naam | VARCHAR2 | Ja |
| Afkorting | VARCHAR2 | Ja |
| Start\_broed | DATE | Ja |
| Eind\_broed | DATE | Ja |
| Punten | NUMBER | Ja |

|  |  |  |
| --- | --- | --- |
| **Soort\_waarneming** | | |
| **Kolom** | **Datatype** | **Verplicht?** |
| ID | NUMBER | Ja |
| Naam | VARCHAR2 | Ja |
| Punten | NUMBER | Ja |

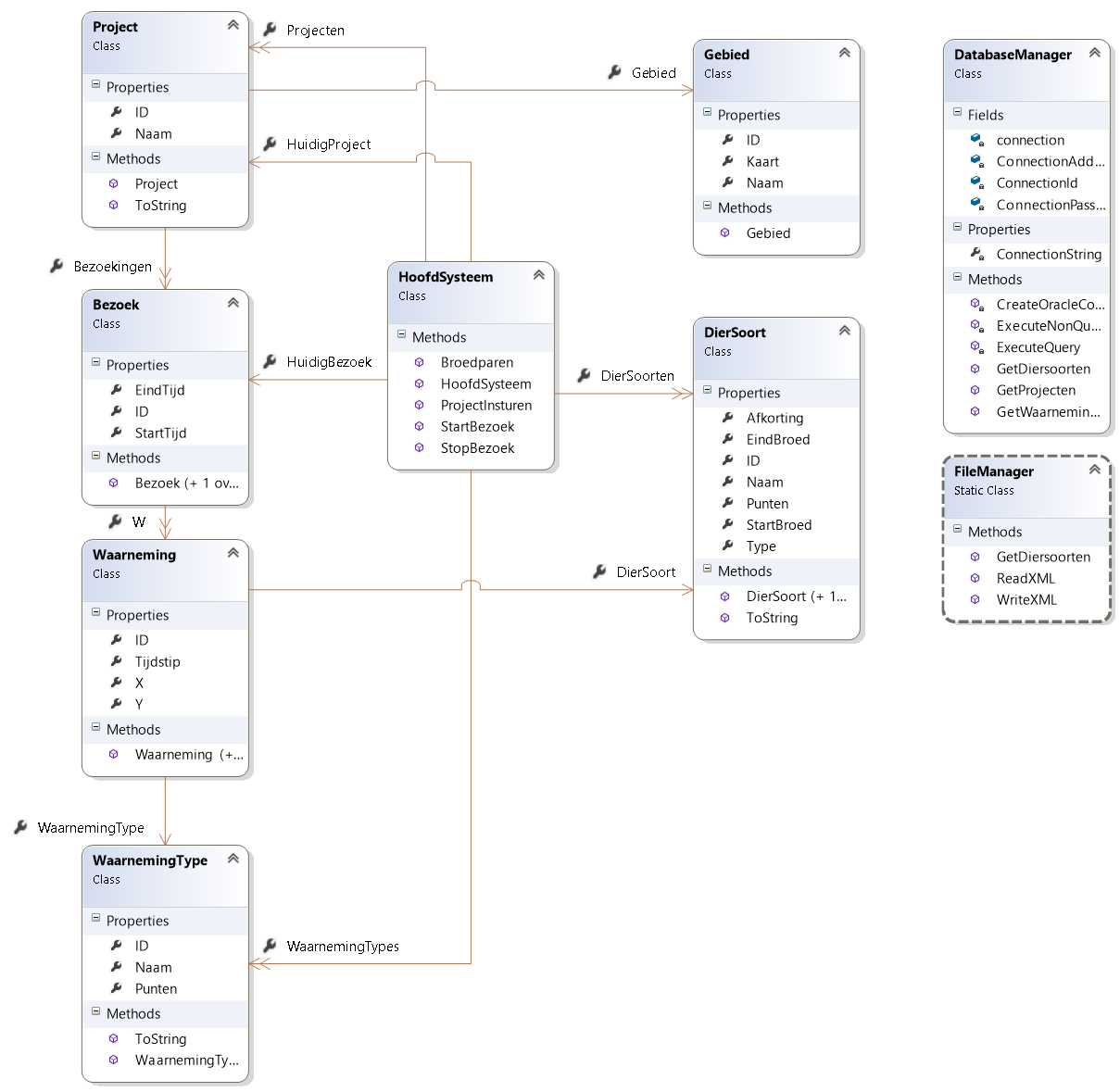
## Klassendiagram

### Domeinmodel

De domeinmodel laat de verschillende relaties tussen de klassen zien, zoals u ziet zijn hierbij nog geen properties of methodes te zien. Hierbij heb ik er voor gekozen om zowel de DatabaseManager, als FileManager een static class te maken. Dit is de reden waarom deze twee geen relaties hebben met de rest van de klassen.

Als laatst uitzonderlijk geval is de directie connectie tussen HoofdSysteem en Bezoek. Dit is vanuit een coderingsstandpunt handiger, hierdoor zal je namelijk niet telkens via project naar je huidige bezoek object toe moeten gaan.

## Implementatie model



In de hierin bovenstaande implementatie model is de uiteindelijke versie van de software klassen te zien. Alle lijsten zijn weergegeven als lijnen(relaties), ook ziet u bij sommige constructors +1 staan. Dit komt vanwege de XML exporteer methode, deze had lege constructors nodig om de objecten om te zetten naar een XML formaat. Aan de hand van deze implementatie model hoop ik u een beeld te geven van hoe de applicatie er uit zou zien.

Er is gekozen voor een huidigbezoek en huidigproject, voor handigheid tijdens coderen. Dit zorgde ervoor dat ik niet telkens via de project op zoek moest gaan naar de bezoek waarmee ik aan het werk was.