

Ontwerp document

Live Performance Guus Hamm



25 juni 2015

Fontys Ict

Inhoud

[Inleiding 2](#_Toc422926288)

[ERD & Database Ontwerp 3](#_Toc422926289)

[ERD 3](#_Toc422926290)

[Database Ontwerp 3](#_Toc422926291)

[Klassendiagram 4](#_Toc422926292)

[Domeinmodel 4](#_Toc422926293)

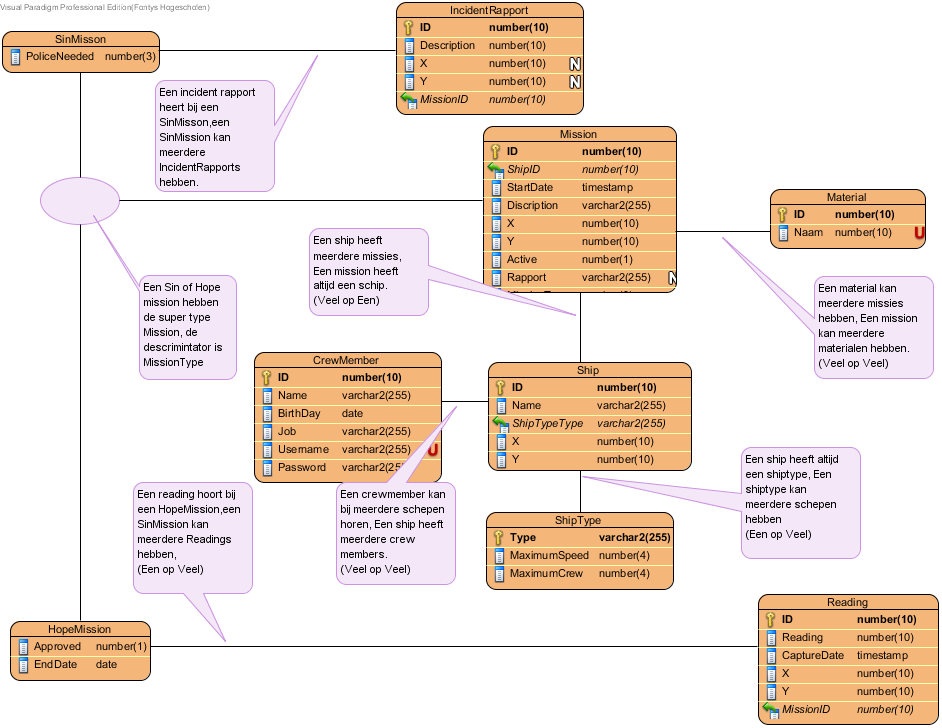
[Implentatiemodel 4](#_Toc422926294)

# Inleiding

Dit document bevat alle ontwerp beslissingen die zijn genomen tijdens de ontwerp fase van het live performance project. Dit document bevat dus het Entiteiten-relatie-diagram, het database ontwerp, het klassediagram en een toelichting op al deze diagrammen en ontwerpen. Door het bestuderen van dit document moet het voor een andere software engineer met C# en Oracle ervaring mogelijk zijn om aan het project te werken en het verder uit te bereiden.

# ERD & Database Ontwerp

## ERD



1 Het erd is gemaakt met Visual Paradigm, Visual Paradigm maakt bij een veel op veel tabel automatisch een koppel tabel en voegt bij een een op veel relatie automatisch een foreign key toe. Omdat dit een erd is en geen database ontwerp staan de relaties geannoteerd.

2 Een N staat voor een Nullable constraint

3 Een U staat voor een Unique constraint

In het ERD staat een mission centraal, alle andere zaken zijn namelijk afhankelijk van een mission. Een mission heeft meerdere subtypen en een discriminator. Als de MissonType een SinMisson is dan krijgt hij de extra attributes van een sin missie, hetzelfde geldt voor als de missie een hope missie is.

Een missie heeft altijd een ship, een shiptype. Een ship heeft een of meerdere crew member(s). Een crew member heeft een functie.

## Database Ontwerp

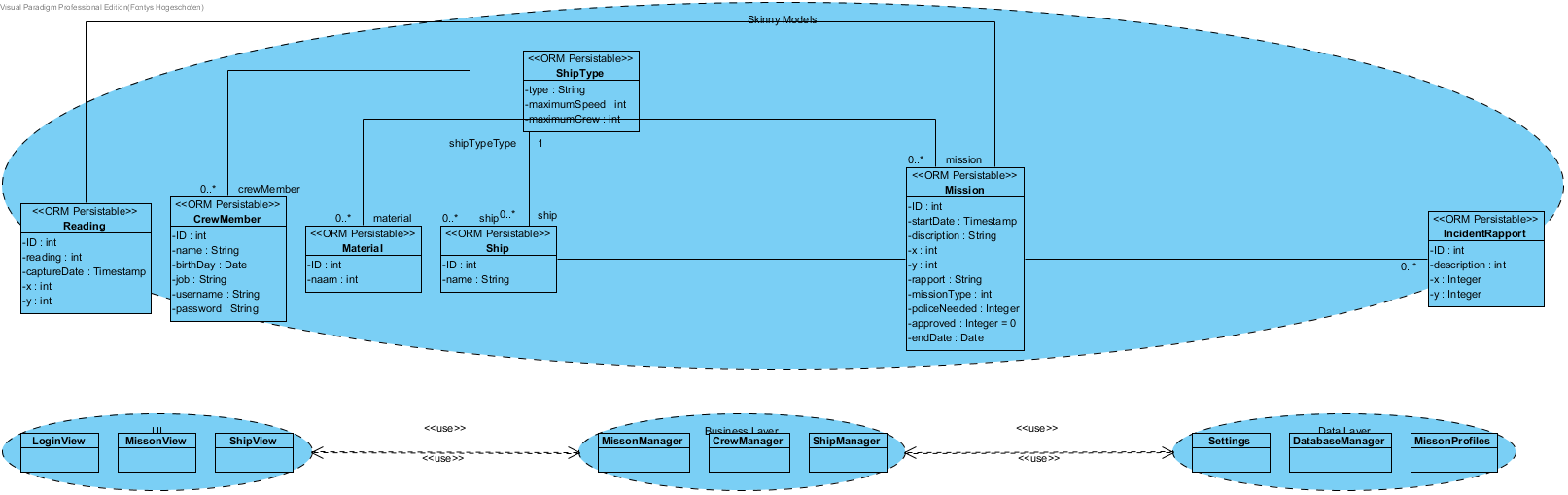


Bij de overgang van ERD naar Database Ontwerp heb ik de volgende handelingen uitgevoerd:

* Bij een op veel relaties is een foreign key toegevoegd aan de veel kant.
* Bij veel op veel relaties is een koppel tabel toegevoegd.
* Alle subtypen tabellen zijn samengevoegd in een tabel met een discriminator.
* De layout van het Database Ontwerp is iets verander ten opzichten van het ERD omdat door de koppeltablen de oude layout niet de optimale verdeling geeft.

# Klassendiagram

## Domeinmodel

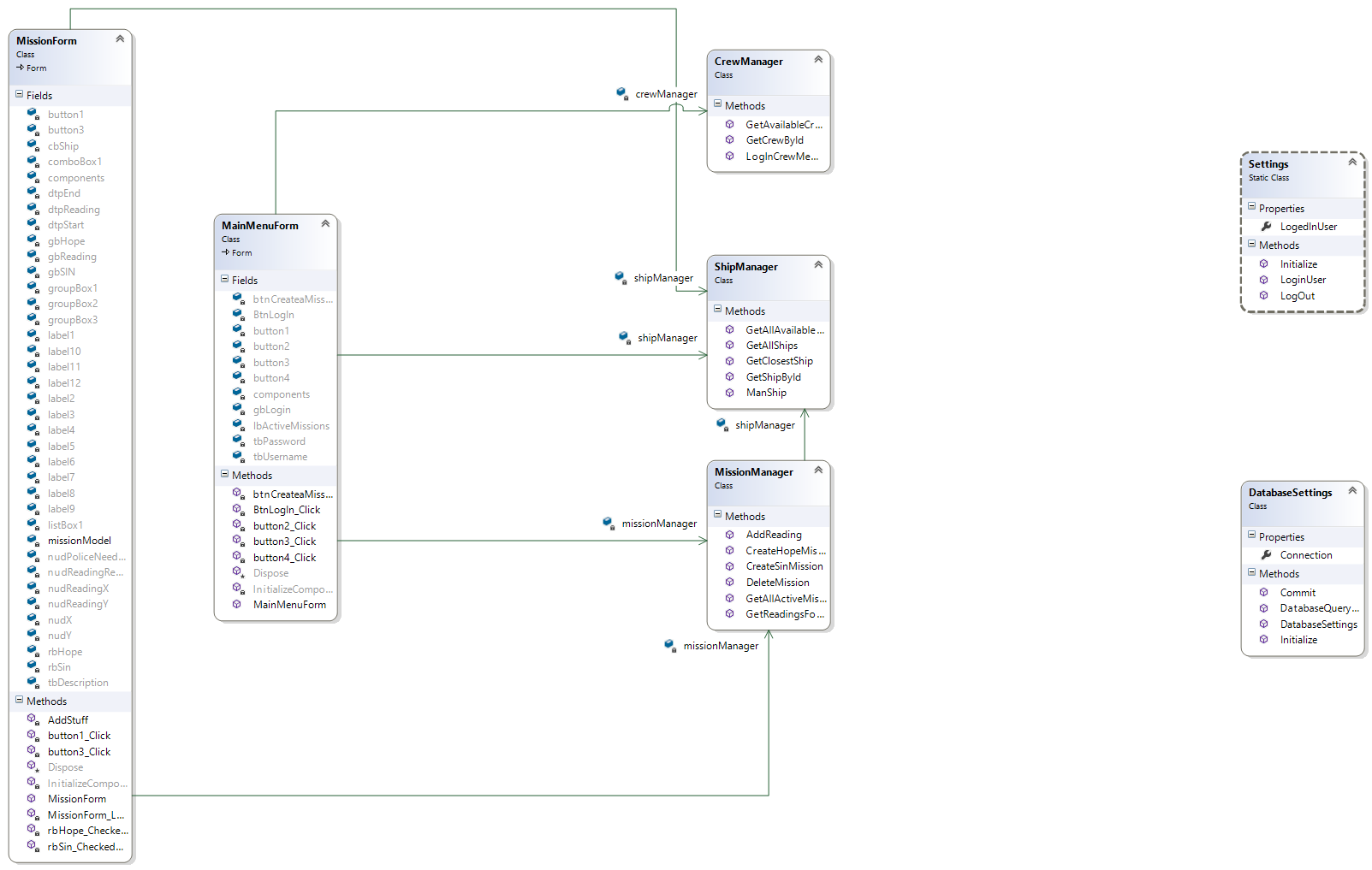


Het klassen diagram bestaat uit een aantal duidelijke lagen:

* De ui laag, wat de gebruiker ziet. De input van de gebruiker wordt gevalideerd en dan door gestuurd naar de logica laag. De ui laag is ook verantwoordelijk voor het weergeven van data die van de logica laag komt in de vorm van models.
* De logica laag, de projectcontroller en de accountcontroller. Hier zit de logica van de applicatie de ui laag stuurt en voert de logica bewerkingen uit en dan stuurt de logica laag een model door naar de datalaag. De logica laag krijgt verder vanuit de ui laag de vraag om data door te sturen vanuit de datalaag, de logica laag stuurt dan een model terug.
* De Datalaag zorgt ervoor dat er gecommuniceerd kan worden met de database, en bevat alle data van de applicatie.

Alle drie de lagen geven aan elkaar data door doormiddel van thin models dit zijn c# objecten die database tabelen vertegenwoordigen. Deze models zorgen ervoor dat de communicatie tussen de verschillende datalagen goed verloopt.

## Implentatiemodel



4 In het bovenstaande klassendiagram zijn de verschillende data lagen goed te zien, de Form klassen vertegenwoordigen de GUI laag, de Manager klassen bevatten de logica laag en de Settings klassen vertegenwoordigen de datalaag. Hier niet afgebeeld maar wel aanwezig zijn de skinny models die database objecten vertegenwoordigen, deze zijn laag doorbrekend.

5 In het bovenstaande klassendiagram zijn er geen lijntejs tussen de settings klassen en de manager klassen dit komt omdat deze klassen static zijn en dus niet te modeleren zijn, maar deze settings worden door de Manager klassen gebruikt.

## Algoritme Flow Chart

