Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 12/11/15 | 1.0 | First version | Mehmet Ali Incekara |
| 28/11/15 | 1.1 | Add SRS ref. | Mehmet Ali Incekara |
| 19/12/15 | 1.2 | Updated all Pictures and Added DB Model | Marc Mahler |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Definitions, Acronyms, and Abbreviations 4

1.3 References 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

4. Use-Case View 4

5. Logical View 4

5.1 Overview 4

5.2 Separated view 5

5.2.1 Model 5

5.2.2 View 6

5.2.3 Control 6

5.2.4 Database 7

6. Process View 7

7. Deployment View 7

8. Implementation View 7

9. Data View 8

10. Size and Performance 8

11. Quality 8

# Introduction

## Purpose

This document provides an architectural overview of Nappy, the ingenious in aspects of different architectural views.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| SAD | Software Architecture Document |
| MVC | Model-View-Controller |
|  |  |

## References

GitHub-Docs: <https://github.com/nappydevelopment/docs>

SRS:

<https://github.com/nappydevelopment/docs/blob/master/pdfs/Software%20Requirements%20Specification.pdf>

# Architectural Representation

The project Nappy, the ingenious will use the MVC-principles.

# Architectural Goals and Constraints

The main goal of this architecture is to separate the view from the logic. The view is “stupid” and knows nothing.

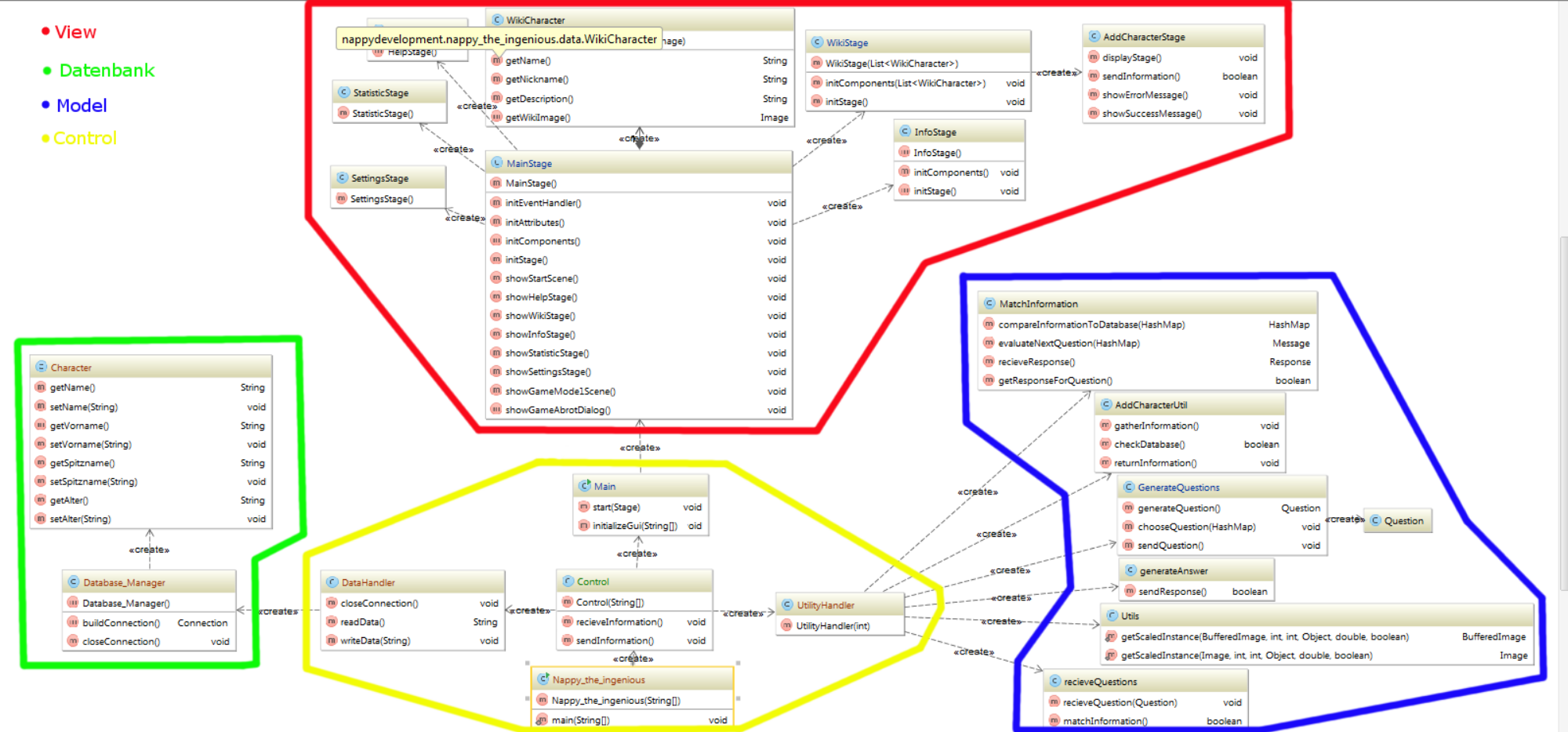
We will use the Framework from JavaFX for our project.

# Use-Case View

(n/a)

# Logical View

## Overview



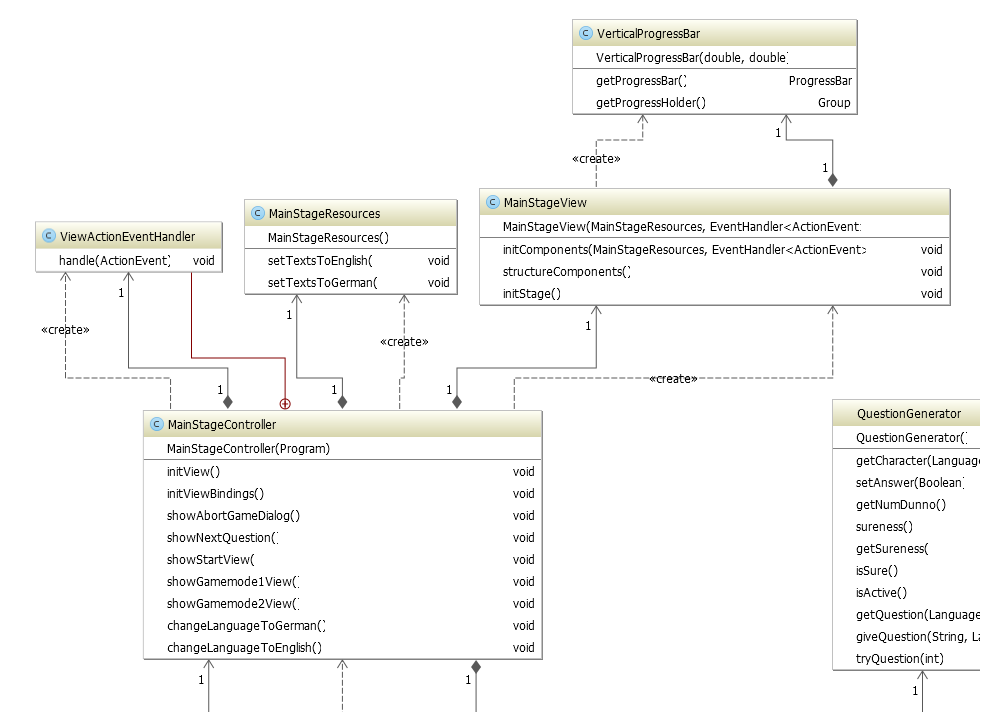
You can also find this picture (a little bit bigger) on GitHub: <https://github.com/nappydevelopment/docs/blob/master/pdfs/Class_Diagramm.pdf>

After we updated our project, there was a new class diagram produced. In the new Class Diagramm we completed our work on the MCV principle and used the JavaFX Framework to create it.

The new Class Diagram is pretty huge, so you won’t be able to read what is in there in this document.

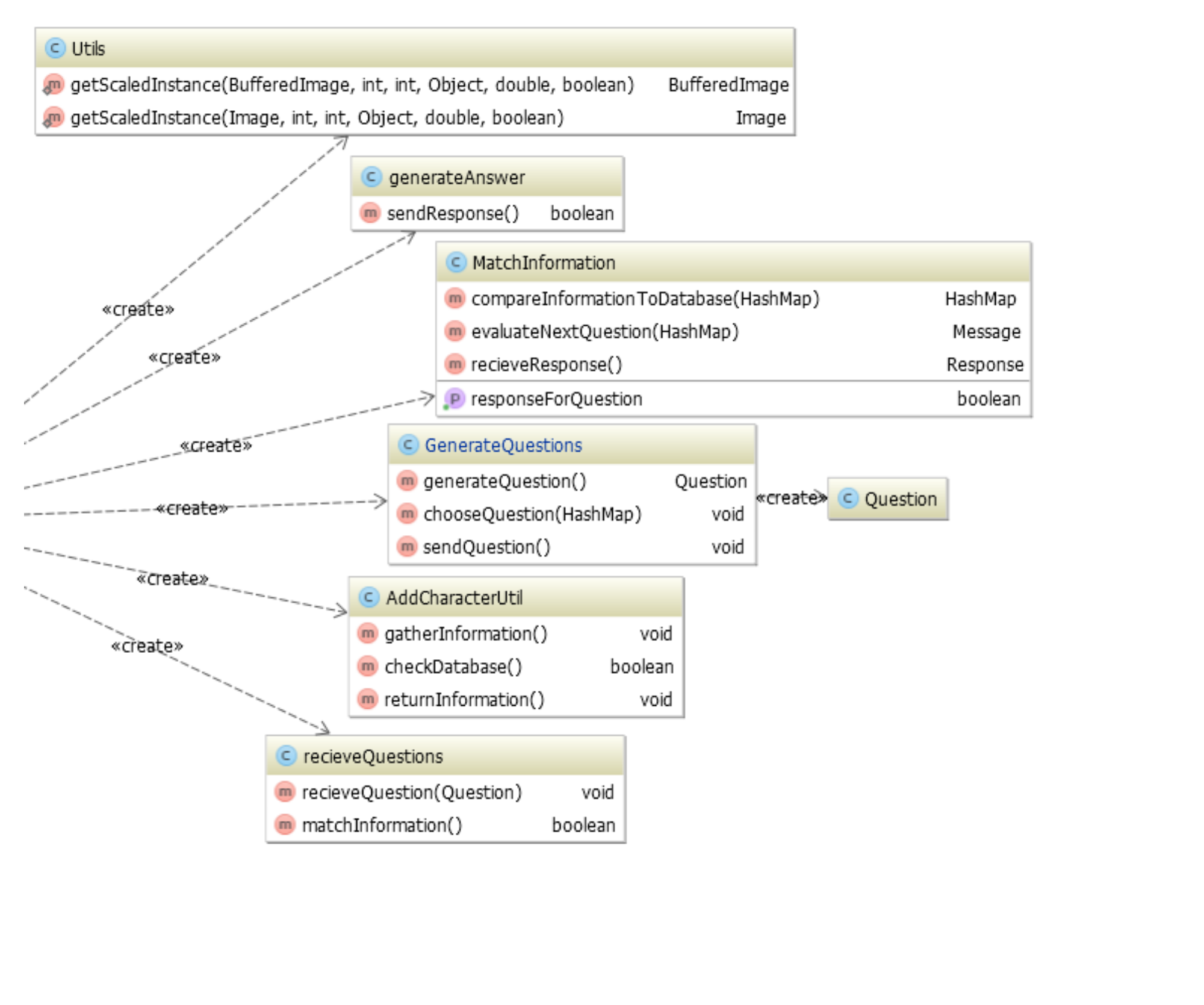
You can follow the provided link to see the full Class Diagram as an SVG and zoom and scroll in it.

<https://github.com/nappydevelopment/docs/blob/master/svg/New%20Classdiagram.svg>

Our MVC concept consists in many “stages” as they are called in JavaFX. A stage represents one window or one label inside a window. Every Stage has their own MVC conept, which makes it pretty hard to show a detailed view about Model, View, Control and Database. That’s why we decided, to add one single stage right here and mention the important parts in the chapters below. 

## Separated view

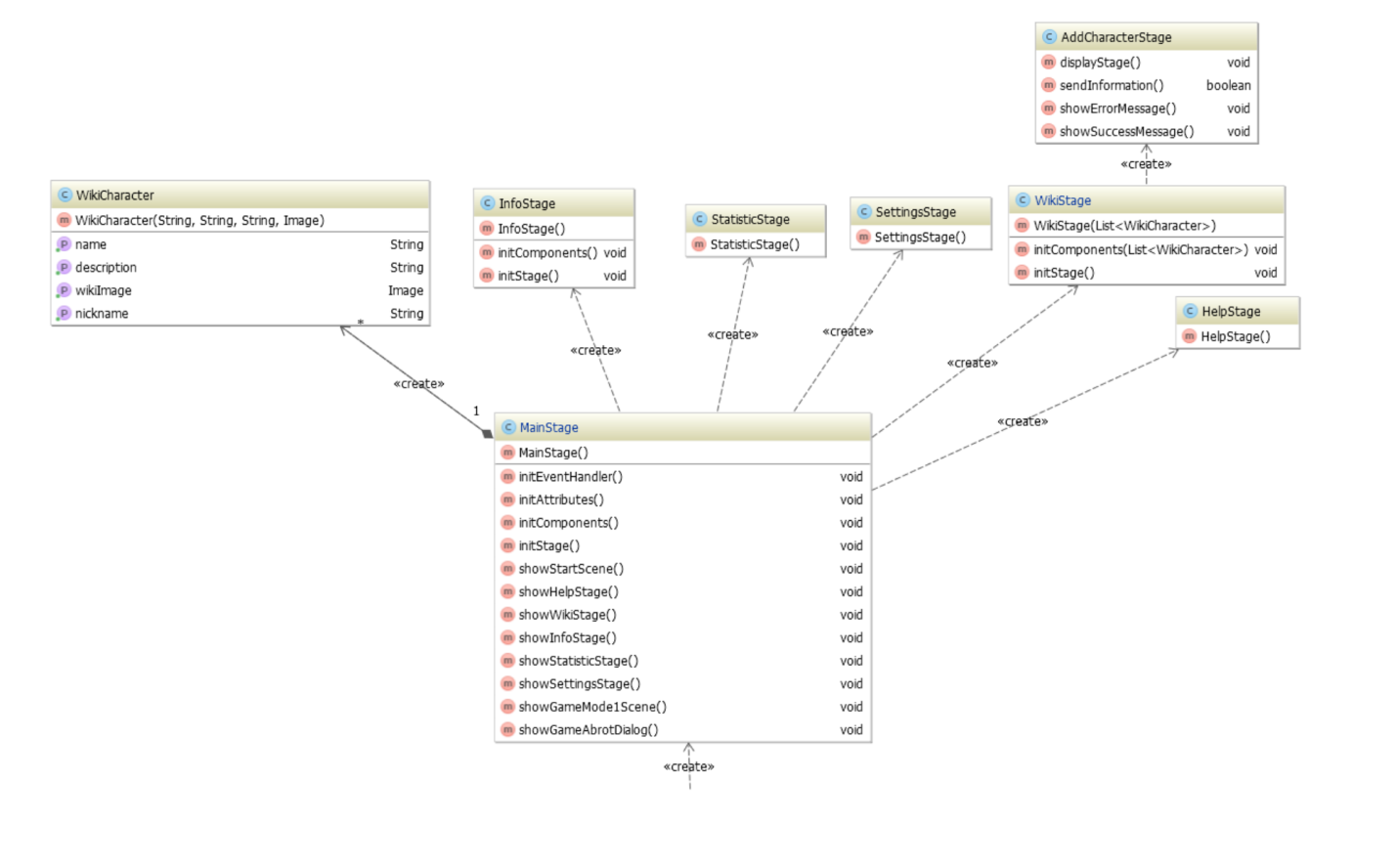
### Model



<https://github.com/nappydevelopment/docs/blob/master/pdfs/Model_Class_Diagramm.pdf>

In the picture in the last chapter (one of our stages), there are two classes, which belong to the Model of this Stage. They are the “ViewActionEventHandler” and the “MainStageResources”

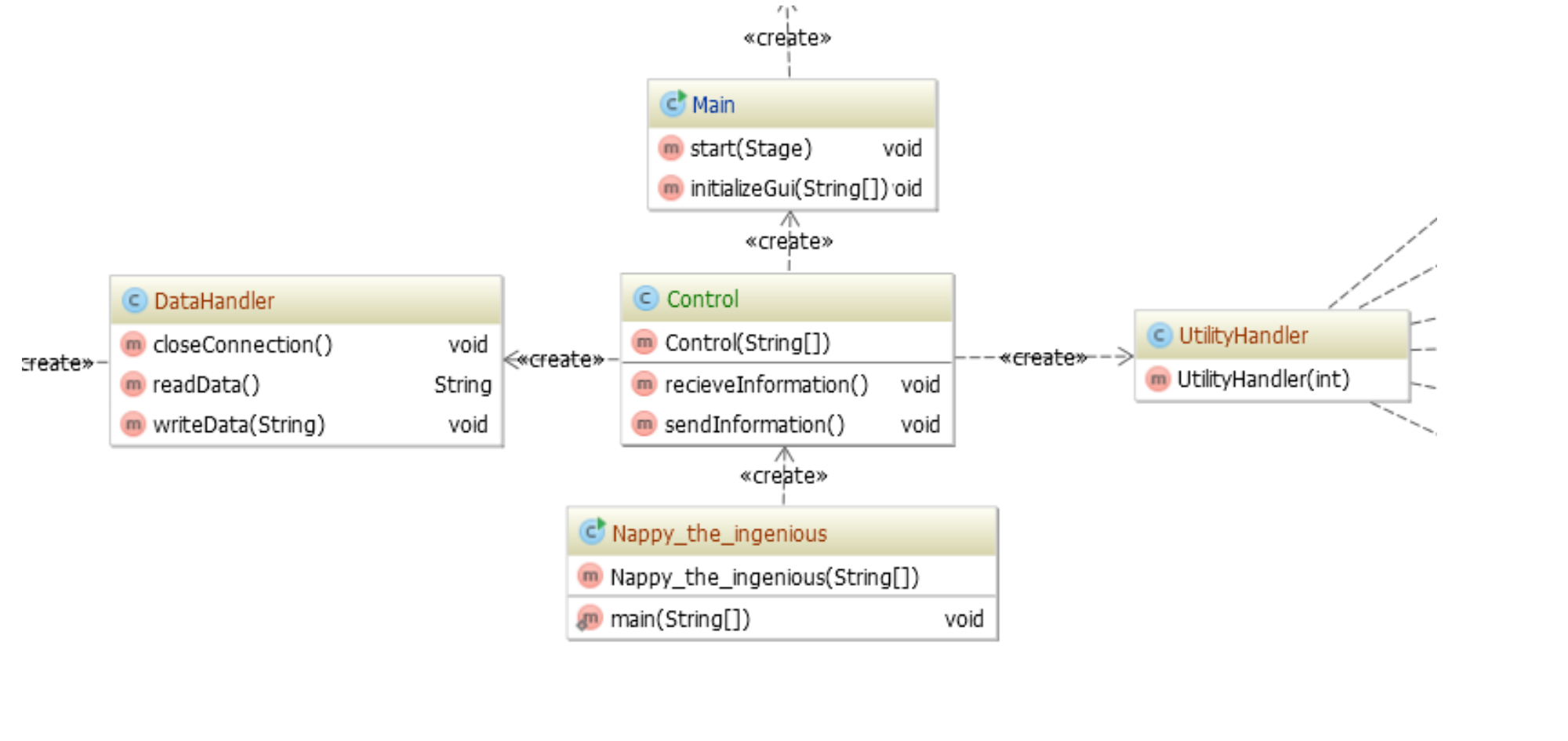
### View



<https://github.com/nappydevelopment/docs/blob/master/pdfs/View_Class_Diagramm.pdf>

The view of our new architecture (picture in the chapter “Overall Diagram” shows two classes, that belong to the View of our MainStage, they are “MainStageView” and “VerticalProgressBar”

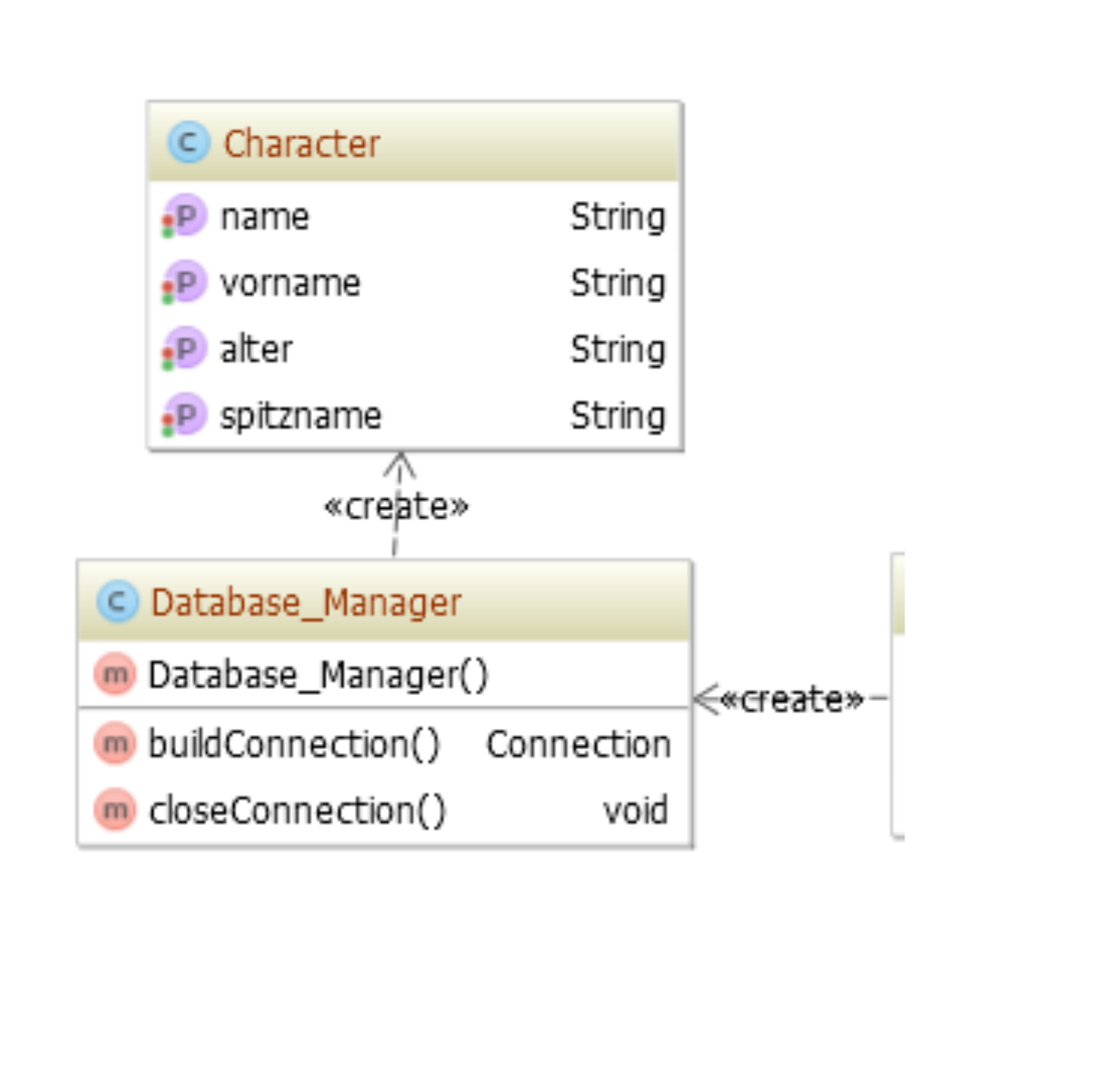
### Control



<https://github.com/nappydevelopment/docs/blob/master/pdfs/Control_Class_Diagramm.pdf>

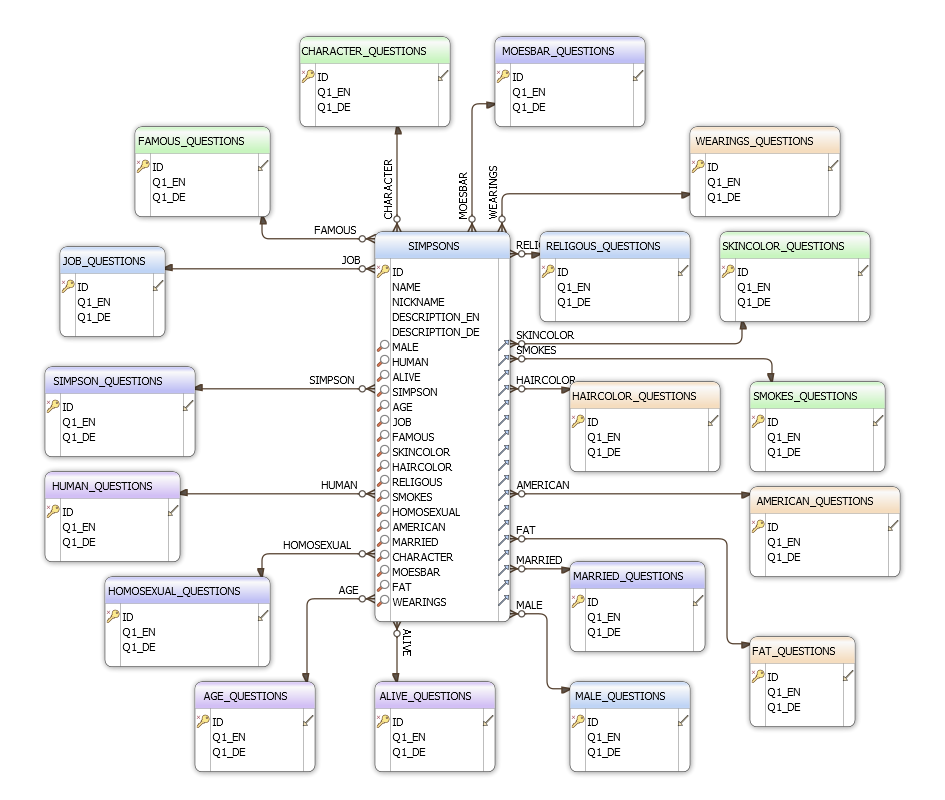
At last, the Contol should be mentioned. In The picture of our MainStage, it is the “MainStageController”.

### Database



<https://github.com/nappydevelopment/docs/blob/master/pdfs/Datenbank_Class_Diagramm.pdf>

For our Database, we also have a model, that might give you a short look at our Database-structure:



<https://github.com/nappydevelopment/docs/blob/master/presentation/Database%20Diagram.svg>

# Process View

(n/a)

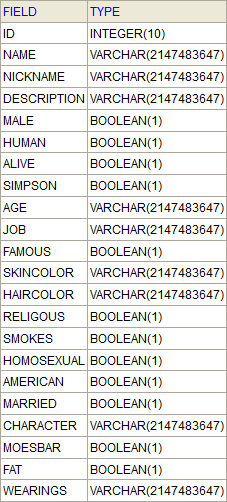
# Deployment View

(n/a)

# Implementation View

(n/a)

# Data View



(tbd)

# Size and Performance

The size and the performance are really important for our project. We are trying to keep the size of the project as small as possible. Also we will load our database into the memory to increase the performance. So we will need a few seconds until the program starts to load all pictures and the database.

# Quality

The quality of our project Nappy, the ingenious is also important. We used a lot of sources to get all information about the characters and so our database has a stable foundation.