

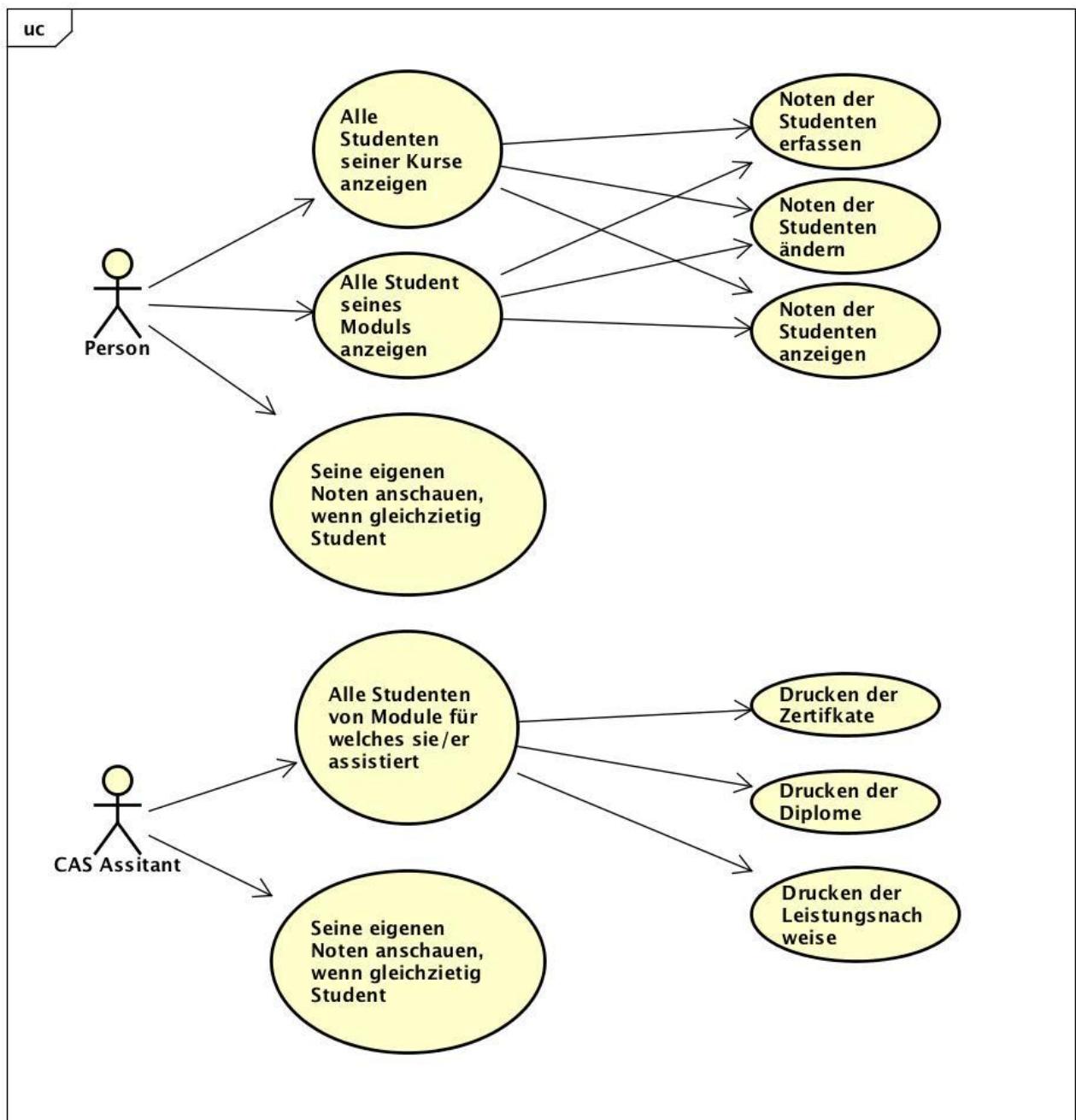
CAS SD: Projekt EqualSD
Datum: 23.03.2017
Autor(en): Sven Bolzern, Christian Etter

Projekt Dokumentation

Projekt Dokumentation	1
1 Use Cases	2
1.1 Diagramm	2
1.2 Beschreibungen	3
2 Datenmodell	4
2.1 ER-Modell	4
2.2 Datenbank-Schema	5
3 Graphische Benutzeroberfläche	6
3.1 Ergonomie-Überlegungen	6
3.2 GUI-Prototyp	9
4 Datenbank Schnittstelle	13
4.1 Klassendiagramm	13
4.2 Design-Überlegungen	14
5 Gesamtdesign	15
5.1 Klassendiagramme	15
5.2 Sequenzdiagramme	16
5.3 Design-Überlegungen	17
5.3.3 XML Format	18
6 Implementation	20
6.1 Zugriffsbeschränkung	20
6.2 Notenberechnung	20
6.3 Dokumenterzeugung	20
7 Erreichte Ziele und gemachte Erfahrungen	21

1 Use Cases

1.1 Diagramm



1.2 Beschreibungen

Um die Lesbarkeit in diesem Dokument zu erleichtern, verzichten wir auf die Aufzählung beider Geschlechter. Es ist aber zu beachten, dass beide Geschlechter gemeint sind: wenn wir ‚er‘ schreiben ist er oder sie gemeint.

Für alle Use Cases gilt, jeder User muss sich zuerst mit einem Login authentifizieren, damit die Daten vor unberechtigtem Zugriff geschützt sind.

Eine Person kann ein Student und/oder ein Dozent und/oder ein CAS Verantwortlicher sein. Jedoch kann ein CAS Verantwortlicher oder ein Dozent nie die Rolle des CAS Assistenten seiner eigenen Kurse/CAS haben.

Wenn die Person ...

- ... ein Student ist, dann kann er nur seine eigenen Noten pro Kurs anzeigen lassen jedoch nicht bearbeiten.
- ... ein Dozent ist, dann kann er alle Noten der Studenten seiner Kurse erfassen, anzeigen und bearbeiten. Falls er gleichzeitig ein Student ist, kann er seine Noten seiner Besuchten Kurse anzeigen aber nicht bearbeiten.
- ... ein CAS Verantwortlicher ist, dann kann er alle Noten der Studenten von ihrem/seinem CAS erfassen, anzeigen und bearbeiten. Falls der Verantwortliche gleichzeitig ein Dozent eines Kurses eines anderen CAS ist, kann er die Noten aller Studenten erfassen, anzeigen und bearbeiten. Falls der Verantwortliche auch noch ein Student ist, dann kann er gleichzeitig seine Noten seiner Besuchten Kurse anzeigen aber nicht bearbeiten.
- ... ein CAS Assistent ist:

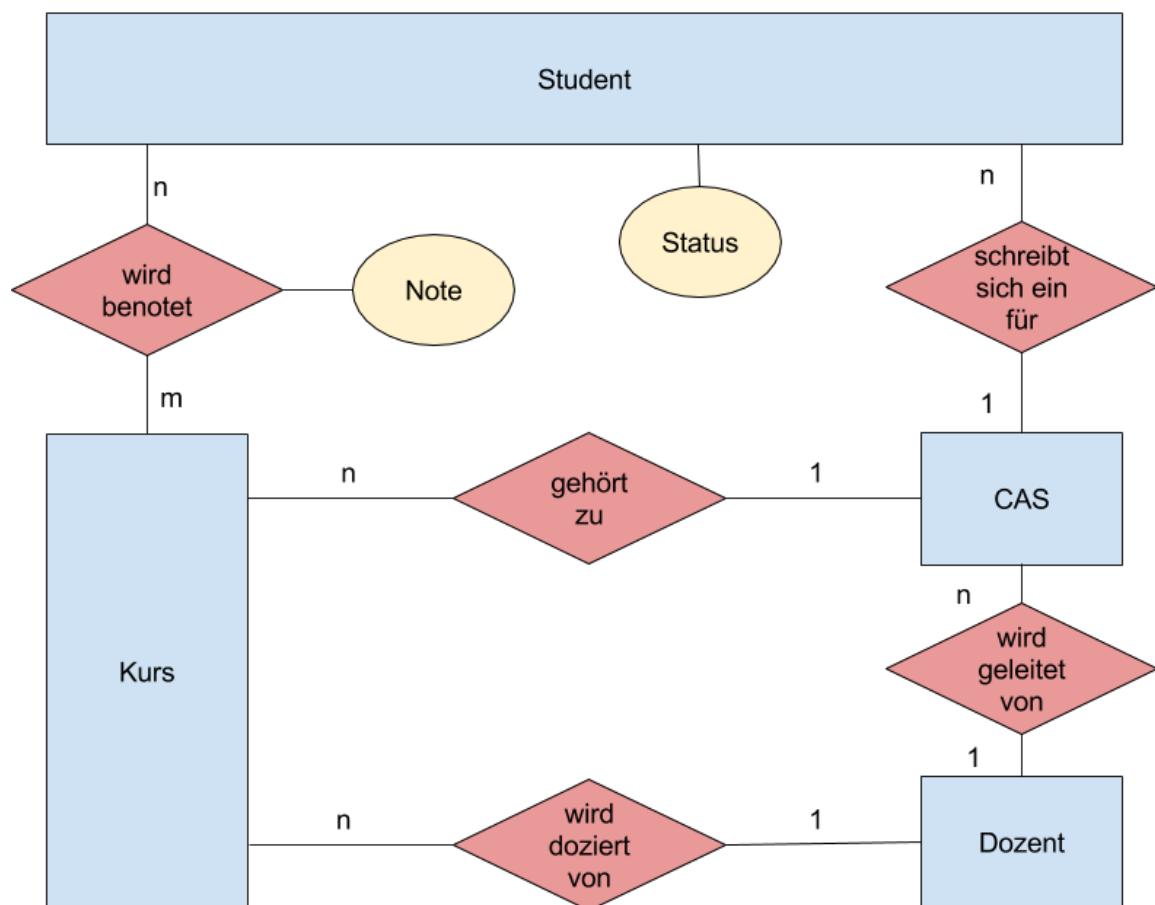
Ein Assistent ...

- ... kann alle Noten der Studenten des CAS, für das er assistiert, anzeigen. Wenn alle Noten vorhanden sind, kann er die Leistungsnachweise und für Studenten mit einem Notendurchschnitt über 50% ein Zertifikat/Diplom drucken. Änderungen der Noten sind für den Assistenten nicht möglich.
- ... kann, wenn er auch ein Student ist, die Noten der besuchten Kurse anzeigen aber nicht bearbeiten.

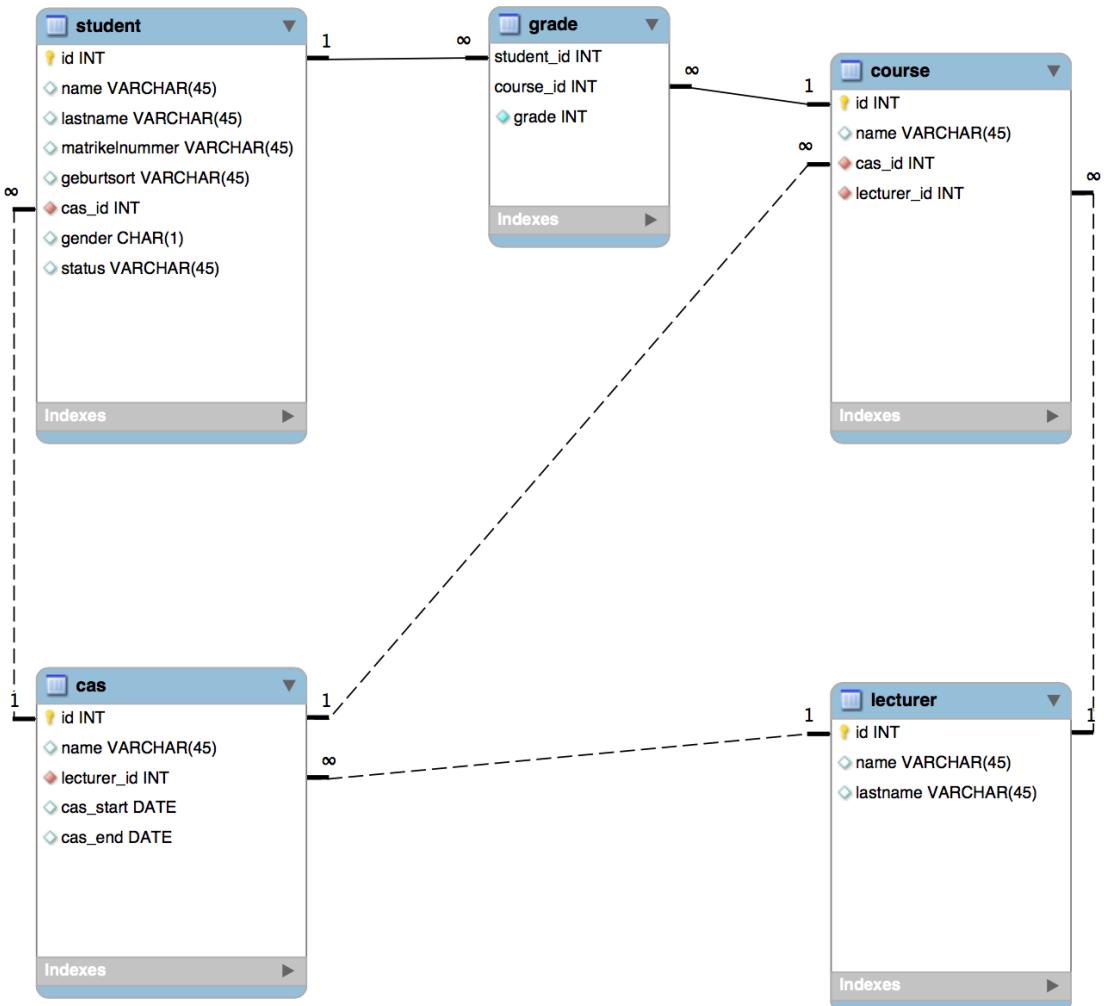
2 Datenmodell

2.1 ER-Modell

Aufgrund unklarer Beschreibung im Anforderungsdokument haben wir angenommen, dass ein Student zur gleichen Zeit nur einem CAS zugeordnet werden kann. Als Folge dieser Annahme macht es auch wenig Sinn, die CAS mehrerer Semester zu persistieren. Dieses Datenmodell bildet folglich jeweils nur eine Momentaufnahme ab.



2.2 Datenbank-Schema



3 Graphische Benutzeroberfläche

3.1 Ergonomie-Überlegungen

Um die Applikation so ergonomisch wie möglich zu machen, wird nur das eingebaut, was auch wirklich vom Endbenutzer gebraucht wird:

1. Gesamterscheinung

Die Applikation muss übersichtlich und aufgeräumt aussehen, damit der User beim Öffnen der Applikation genau sieht, wo er was machen kann. Deshalb werden auf den einzelnen Views nur die Daten des "neuesten" Semesters als Default angezeigt. Über ein Dropdown...



...können die anderen erfassten Semester ebenfalls ausgewählt und eingesehen werden. Idealerweise sollten die alten Semester ReadOnly sein, dies wurde aber in unserer Applikation aus zeittechnischen Gründen nicht beachtet.

2. Bedienung

Um dem Benutzer das Bedienen der App zu erleichtern, wurde Internationalization (I18N) in Englisch und Deutsch eingebaut. Verschiedene Einstellungen des Benutzers werden automatisch in einem Properties File gespeichert. Dies beinhaltet die Grösse des Fensters, ausgewählte Sprache und den "PDF Outputpath" den man zuletzt ausgewählt hat.

3. Views

Die Views wurden so designed, dass für jede Rolle eine separate View zur Verfügung steht. So ist für jeden nur das ersichtlich, was für ihn auch wichtig ist. Wir haben zusätzlich zum Login 4 verschiedene Views für die Rollen Student, Teacher, CAS Assistant und CAS Responsible erstellt.

Für die Assistenten werden die einzelnen Studenten eingefärbt. So ist schnell ersichtlich:

1. Ob der Student einen genügenden Notendurchschnitt hat (Grün)
2. Ob eine Note fehlt(orange)
3. Ob zwar alle Noten vorhanden sind, der Notendurchschnitt jedoch ungenügend ist (Weiss)

Modul Requirements Engineering

Student	RE	UML	SE	ProjMgmt	IntProj
Adam Essler	91	58	91	98	91
Alain Kindler	91	71	91	69	91
Christian Hottinger	79	91	67	91	55
Christian Pasche	91	88	91	77	91
Dominik Van der Moolen	99	91	55		77
Emmanuel Cladiden	91	58	91	97	91
Johannes Bois	49	91	79	91	75
Marcel Hard	78	91	65	91	52
Matthias Maple	51	91	89	91	78
Pascal Bonhote	91	87	91	79	91
Stefan Falcon	88	91	78	91	68
Stefan Optiver	91	55	91	66	91
Stephan Allop	1	1	2	91	75
Thomas Amro	91	52	91	59	91

Die Kurse wurden nach Rolle markiert, so hat ein CAS Responsible ein anderes Symbol als ein Teacher, Assistent oder Student. So ist schnell ersichtlich was man für eine Rolle hat:

Beispiel Assistent und gleichzeitig Student:



Software Development
SD-FS16

Requirements Engineering
RE-FS16

Oder CAS Responsible und Teacher:



Software Development
SD-FS16

Java
Java
Datenstrukturen und Algorithmen AlgoData
Relationale Datenbanken und SQL RDB
XML Technologien XML
GUI/Ergonomie GUI
Projekt Proj

Wenn mehrere Kurse pro CAS angezeigt werden, wird auf der obersten Auswahl, also auf dem CAS, immer eine Übersicht über alle Studenten und Kurse gezeigt. Mit dem Klick auf einen spezifischen Kurs wird die Auswahl dann auf den entsprechenden Kurs reduziert.

Die Noten können direkt in der Übersicht mit einem Doppelklick bearbeitet und mit dem Speichern Button gespeichert werden.

Aus ergonomischen Gründen wurde hier explizit ein Speichern Button gemacht, damit keine Missverständnisse entstehen ob jetzt gespeichert ist oder nicht.

Bevor gespeichert wird, sieht man neben dem Speichern Button auch wie viele Noten

bearbeitet wurden. Dies erleichtert es, Änderungen zu erkennen und zu speichern, bevor man die App schliesst.

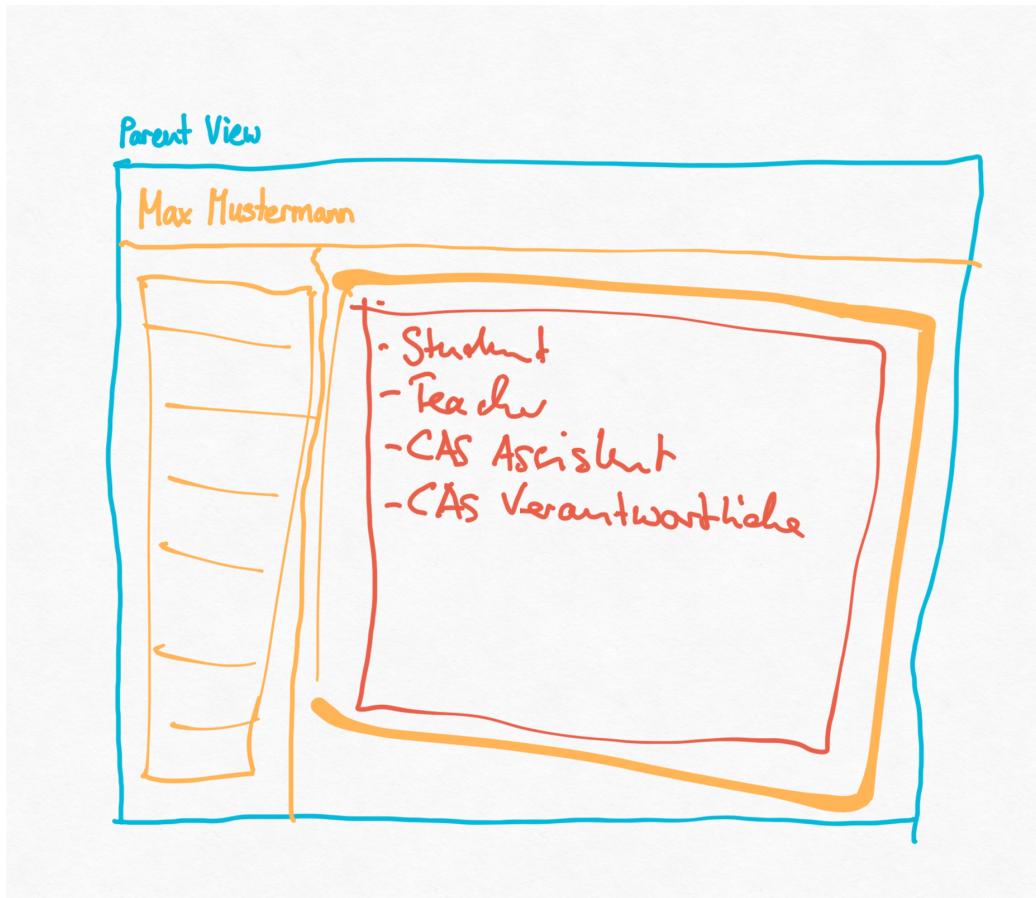
Aus zeittechnischen Gründen wurde darauf verzichtet, den Benutzer auf ungespeicherte Änderungen hinzuweisen, wenn die aktuelle Ansicht verlassen wird oder die App geschlossen wird.

4. Dokumente drucken

Sobald die Noten der Studenten erfasst sind, kann der Assistent die Leistungsnachweise und Zertifikate drucken. Damit dies ergonomischen Ansprüchen genügt, wurde das Generieren der Dokumente so gemacht, dass der Assistent so wenig wie möglich machen muss, es aber trotzdem nachvollziehbar bleibt. Wie in den Views unter Punkt 2. beschrieben, werden die Student eingefärbt, so kann der Assistent nach dem Drucken prüfen kann, ob auch alle Dokumente gedruckt wurden. Er muss sich auch keine Gedanken machen, wer welchen Durchschnitt hat. Sobald er auf "Drucken" klickt, eruiert das Programm, wer alles in dem CAS ist, wessen Noten vollständigen sind und wessen Durchschnitt über 50% ist. Entsprechend der Auswahl werden dann die Dokumente erstellt und im Standard PDF Reader des Rechners geöffnet.

3.2 GUI-Prototyp

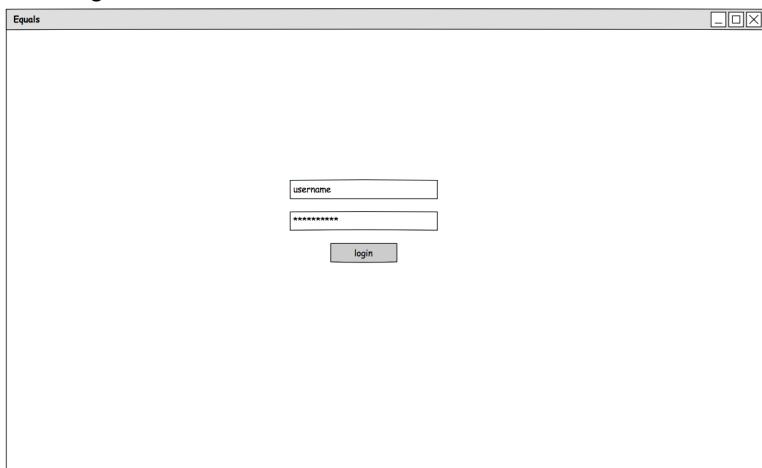
Als erstes haben wir uns überlegt, wie das GUI überhaupt genau aussehen soll. Was man braucht, was praktisch wäre und wie wir sie auseinanderhalten können. Daraus entstand ein erster handgeschriebener Entwurf, wie wir das GUI machen wollten:



Hier waren schon die Aufteilung und die Unterteilung bekannt. Der obere Teil ist reserviert für den Namen der Person, die aktuell eingeloggt ist. Die linke Spalte wurde für die Liste der Kurse oder CAS und Kurse, je nach Rolle, definiert. Der restliche Teil ist für das Anzeigen der Kursdaten vorgesehen. Aus diesen Überlegungen haben wir für jede View einen Prototypen in Form eines Mockups gemacht.

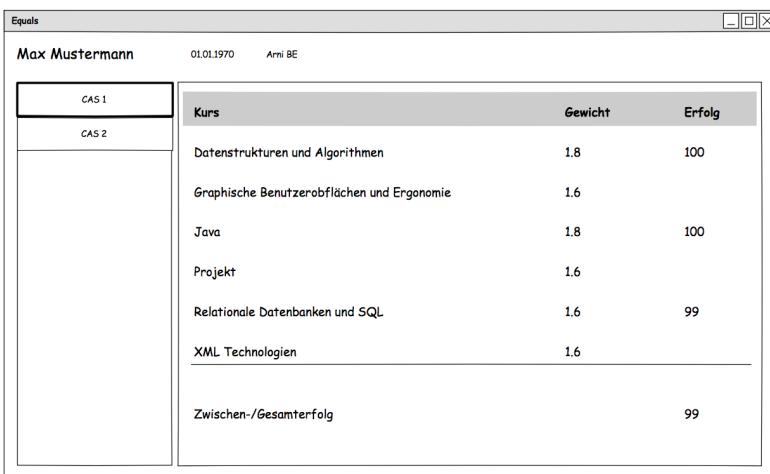
Prototypen Mockups:

User Login:



A window titled "Equals" containing a login form. It has two input fields labeled "username" and "password", and a "login" button.

Studenten Ansicht:



A window titled "Equals" showing student information for "Max Mustermann". It includes fields for "CAS 1" and "CAS 2", and displays a table of course grades.

Kurs	Gewicht	Erfolg
Datenstrukturen und Algorithmen	1.8	100
Graphische Benutzeroberflächen und Ergonomie	1.6	
Java	1.8	100
Projekt	1.6	
Relationale Datenbanken und SQL	1.6	99
XML Technologien	1.6	
Zwischen-/Gesamterfolg		99

CAS Assistentin Ansicht:

Equals

Catherine Flowervalley

Student	AlgoData	Java	Gärtner für Anfänger	XML	Drucken
	Beatrix Amhein	Stephan Fischli	Anfänger	Beatrix Amhein	
Max Mustermann	100	100	68	100	<input checked="" type="checkbox"/>
Anja Naja	50	60	74	65	<input checked="" type="checkbox"/>
Anette Böse	75				<input type="checkbox"/>
Thomas Flow	83		100		<input type="checkbox"/>
Linus Torvalds	32				<input type="checkbox"/>

Dozenten Ansicht:

Equals

Prof. Dr. Pauker

Student	Erfolg
Max Mustermann	100 (100%)
Anja Naja	50 (50%)
Anette Böse	85 (85%)
Thomas Flow	---
Linus Torvalds	32 (32%)

Ansicht eines Dozenten bei einem Kurs, welchen er doziert.

CAS Verantwortlicher Ansicht:

Equals

Prof. Dr. Pauker

Student	AlgoData	Java	Gärtner für Anfänger	XML
	CAS 1	CAS 2	CAS 3	CAS 4
Max Mustermann	100	100	68	100
Anja Naja	50	100	68	100
Anette Böse	---	---	---	---
Thomas Flow	---	---	100	---
Linus Torvalds	32	---	---	---

Ansicht eines CAS-Verantwortlichen bei seinem "eigenen" CAS.

Dozent/Studenten Ansicht:

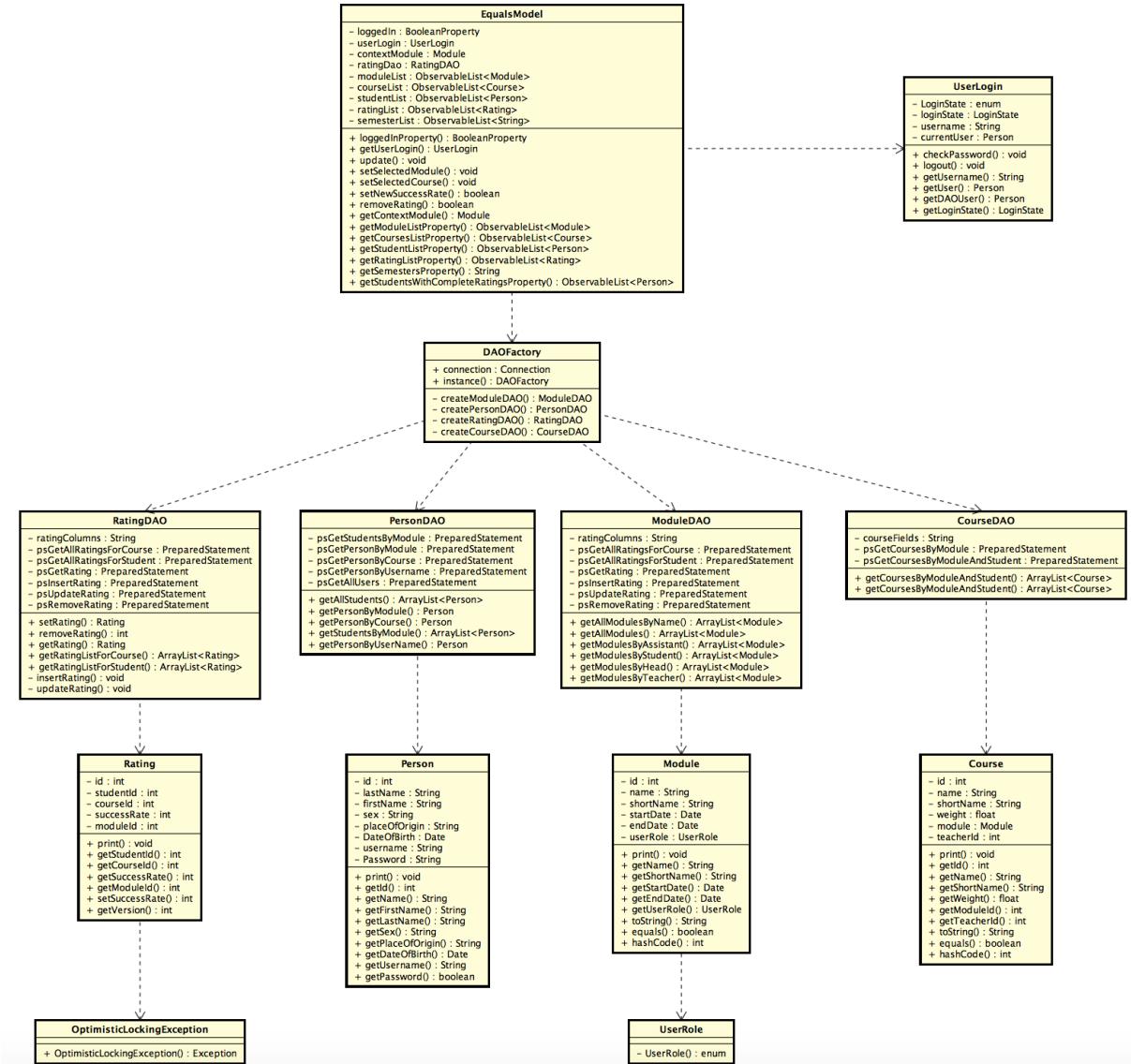
Equals

Prof. Dr. Pauker

Kurs	Gewicht	Erfolg
Datenstrukturen und Algorithmen	1.8	100
Graphische Benutzeroberflächen und Ergonomie	1.6	
Java	1.8	100
Projekt	1.6	
Relationale Datenbanken und SQL	1.6	99
XML Technologien	1.6	
Zwischen-/Gesamterfolg		99

4 Datenbank Schnittstelle

4.1 Klassendiagramm

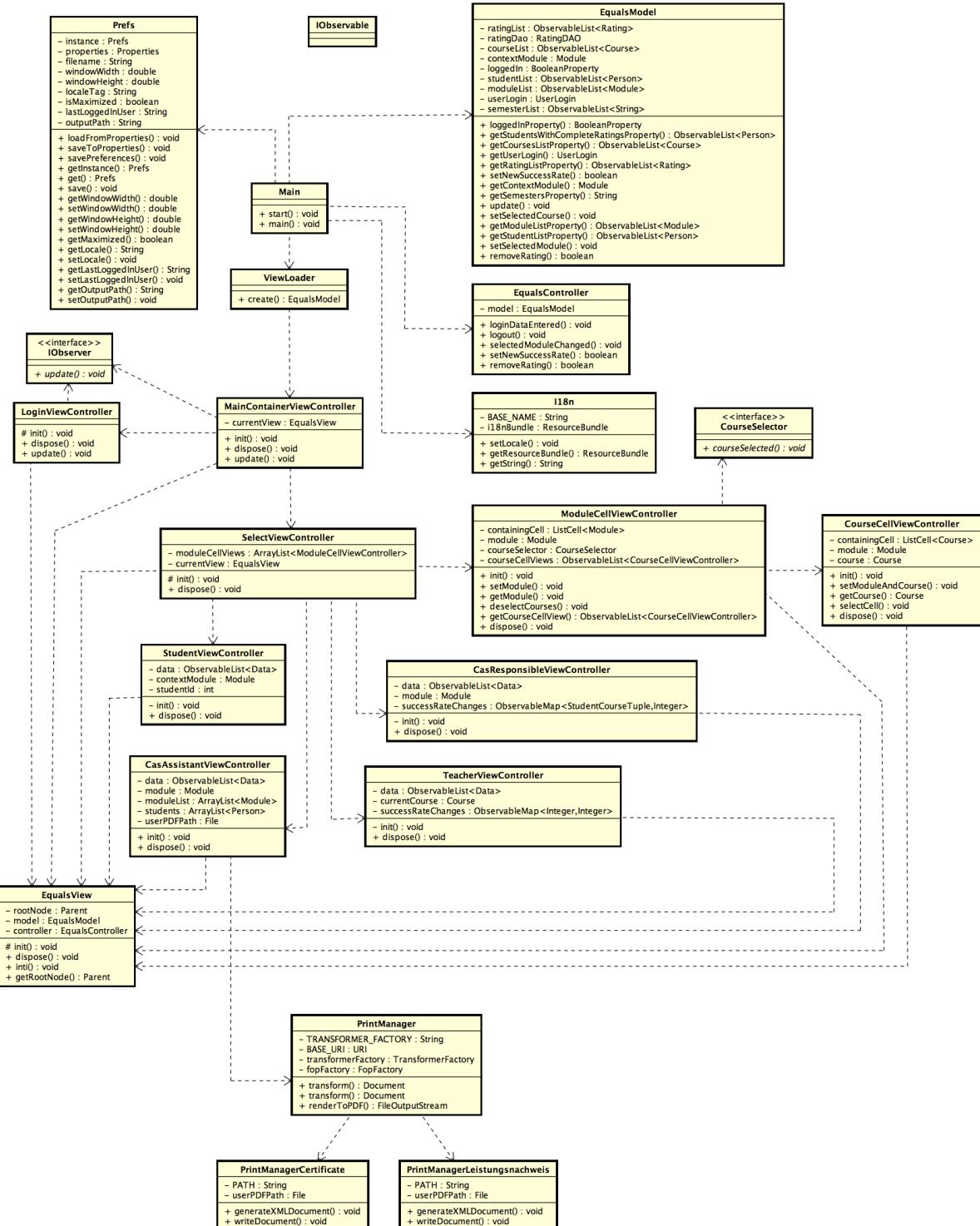


4.2 Design-Überlegungen

Das Design für die Datenbankschnittstelle wurde so gewählt, dass jedes Objekt einzeln oder in Gruppen abrufbar ist. Falls die Applikation in der Zukunft erweitert wird, gibt es keine Einschränkungen und das DAO muss nicht erweitert werden, um spezifische Daten aus der Datenbank abzurufen.

5 Gesamtdesign

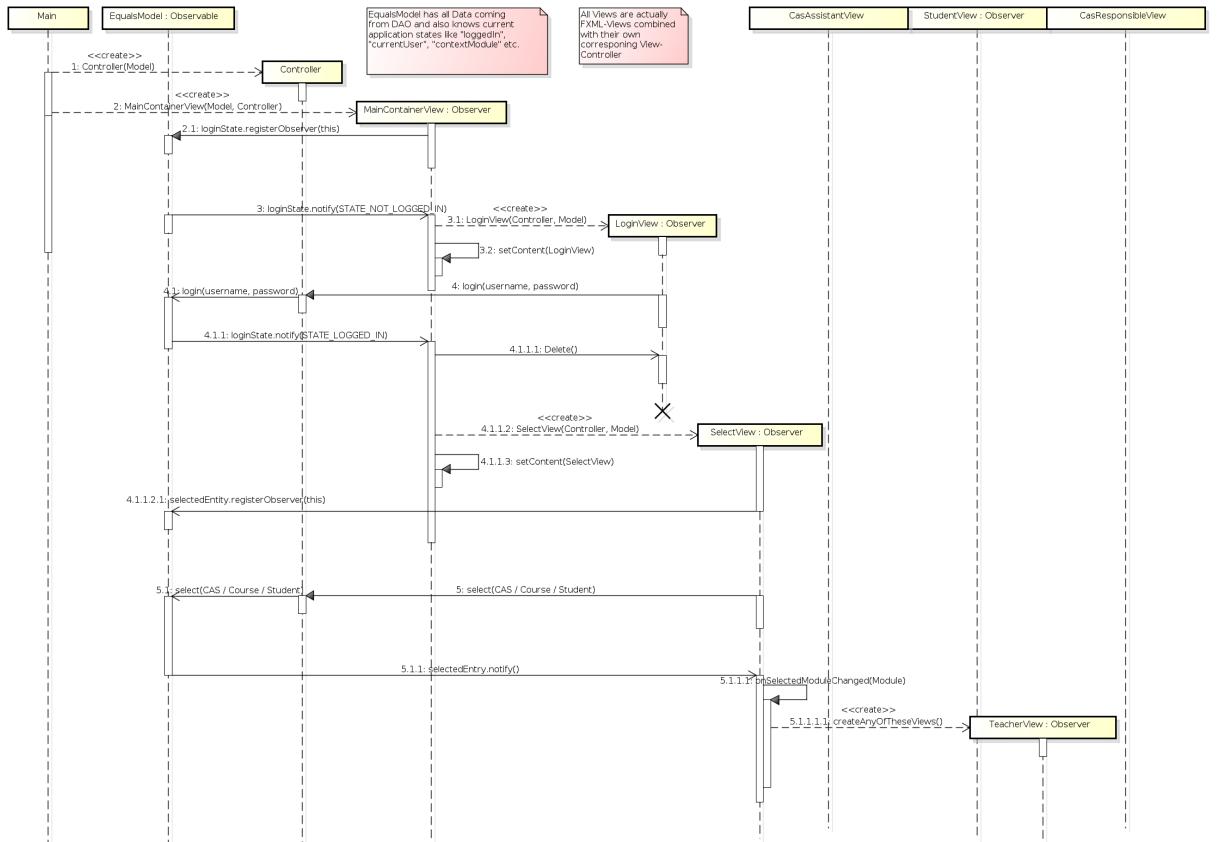
5.1 Klassendiagramme



5.2 Sequenzdiagramme

5.2.1 Konzept der EqualsViews

Das folgende Sequenzdiagramm stellt nur das Konzept dar, wie die verschiedenen Views erzeugt werden. Die Namen der Klassen und der einzelnen Funktionen müssen dabei aber nicht genau mit der Implementierung übereinstimmen.



Hierbei ist zu sehen, dass je nach Applikations-State (Eingeloggt, Modul gewählt, etc.) eine andere View instanziert wird. Was in diesem Diagramm wiederum nicht zu sehen ist, ist wie die Views "geschachtelt" werden, d.h. dass eine View eine andere View beinhaltet kann.

5.3 Design-Überlegungen

5.3.1 Exception Handling

Das Exception Handling wurde soweit gemacht, dass Exceptions abgefangen, aber nicht weiterbearbeitet werden. Unsere Idee wäre, dass wenn eine Exception auftaucht, das Programm ein Popup öffnet und eine entsprechende Fehlermeldung präsentiert, damit der User auch genau weiß, was jetzt schiefgegangen ist und entsprechend handeln kann. Ohne diese Fehlermeldung ist eine Fehlersuche eher schwierig bis unmöglich.

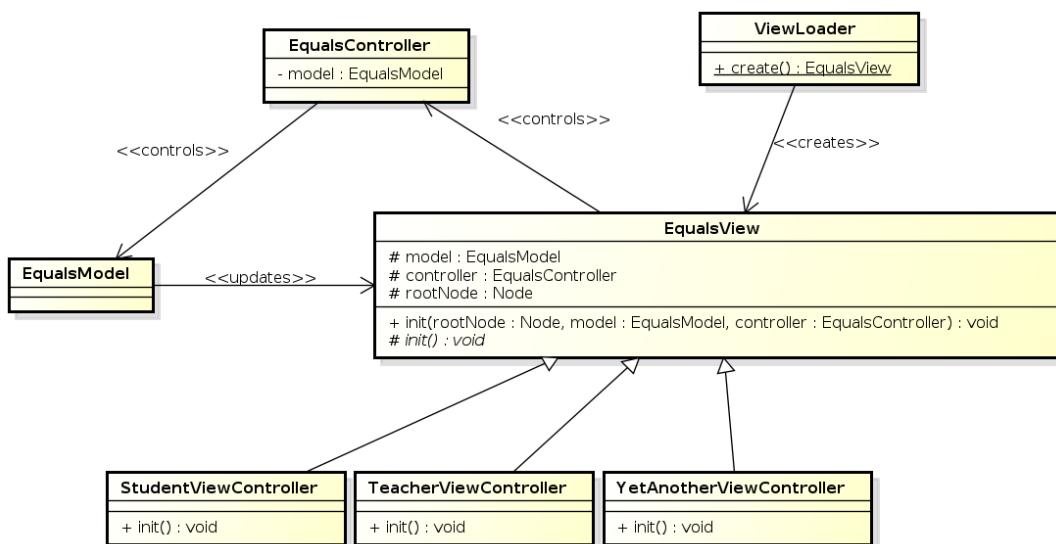
5.3.2 Viewloader

Da das EQUALS-GUI mehrere Zustände haben kann und entsprechend der Zustände verschiedene Inhalte präsentiert (Login, Studentenansicht, Lehreransicht, ...), wollten wir es uns möglichst einfach machen, verschiedene "Views" zu erstellen und zu laden. Daher kam das Konzept der EqualsViews und des ViewLoaders ins Spiel:

Jede View im EQUALS-Projekt ist eine FXML-View mit einem eigenen ViewController, welcher wiederum von der Klasse "EqualsView" erbt. Die EqualsView Klasse enthält 3 Felder, die jede View haben soll: je eine Referenz auf das Modell (EqualsModel), auf den Controller (EqualsController) und auf den root-Node der View selbst.

Dies erlaubt es nun der Klasse "ViewLoader", mit der statischen Methode "create" eine EqualsView zu laden und gleich mit den 3 Feldern zu initialisieren.

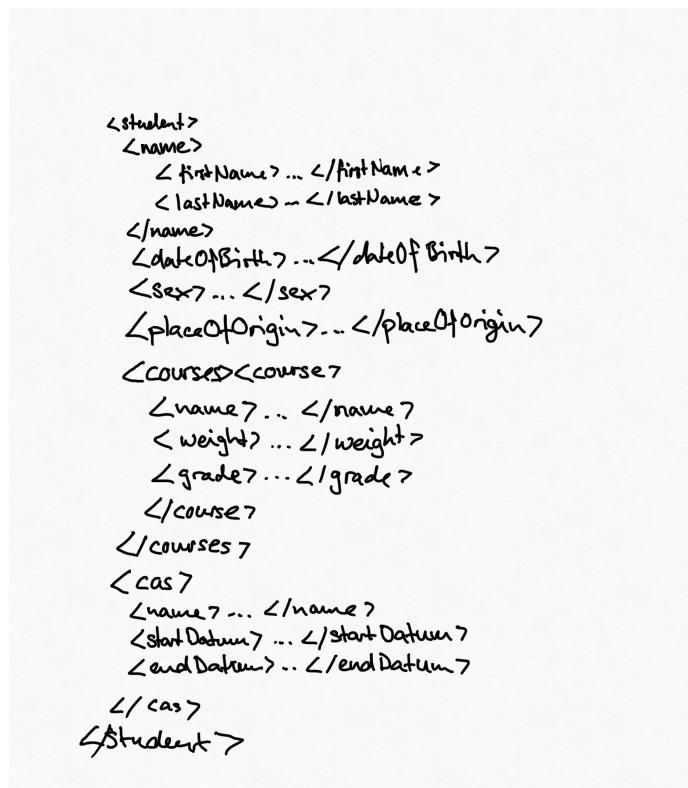
Das nachfolgende Klassendiagramm soll das Konzept erläutern, wobei auch hier der Einfachheit halber nicht jedes Detail mit der Implementierung übereinstimmt.



5.3.3 XML Format

Um die Zertifikate/Diplome und Leistungsnachweise drucken zu können, müssen die erfassten Daten aus der Datenbank gelesen und in ein anderes Format transformiert werden. Dies wurde mit Hilfe von DOM gemacht. Mit DOM kann sehr schnell und einfach ein XML Dokument mit Nutzdaten aus der Datenbank erstellt werden.

Eine erste Skizze des XML Formats handgeschrieben:



The image shows a handwritten XML schema sketch on a light gray background. The schema starts with a root element <student>, which contains a <name> element. Inside <name> are <firstName> and <lastName>. Following <name> are <dateOfBirth> and <sex>. Then there is a <placeOfBirth> element. Next is a <courses> element, which contains multiple <course> elements. Each <course> has <name>, <weight>, and <grade>. After <courses> is a <cas> element, which contains <name>, <startDatum>, and <endDatum>. Finally, there is another <courses> element. The entire structure ends with a closing tag </student>.

```
<student>
  <name>
    <firstName> ... </firstName>
    <lastName> ... </lastName>
  </name>
  <dateOfBirth> ... </dateOfBirth>
  <sex> ... </sex>
  <placeOfBirth> ... </placeOfBirth>
  <courses><course>
    <name> ... </name>
    <weight> ... </weight>
    <grade> ... </grade>
  </course>
  </courses>
  <cas>
    <name> ... </name>
    <startDatum> ... </startDatum>
    <endDatum> ... </endDatum>
  </cas>
</student>
```

Dieses Format hat sich während der Entwicklung noch ein wenig verändert. Das aktuelle Format sieht wie folgt aus:

```
<students>
  <student>
    <firstName>Alex</firstName>
    <lastName>Aargauer</lastName>
    <dateOfBirth>1966-09-18</dateOfBirth>
    <placeOfBirth>Frankfurt am Main</placeOfBirth>
    <sex>m</sex>
    <shortName>aua1</shortName>
    <courses>
      <course>
        <name>Java</name>
        <weight>1.8</weight>
        <rating>91</rating>
      </course>
    <courses>
      <course>
        <name>Datenstrukturen und Algorithmen</name>
        <weight>1.8</weight>
        <rating>89</rating>
      </course>
    <courses>
      <course>
        <name>Relationale Datenbanken und SQL</name>
```

```

<weight>1.6</weight>
<rating>87</rating>
</courses>
<courses>
  <name>XML Technologien</name>
  <weight>1.6</weight>
  <rating>76</rating>
</courses>
<courses>
  <name>GUI/Ergonomie</name>
  <weight>1.6</weight>
  <rating>94</rating>
</courses>
<courses>
  <name>Projekt</name>
  <weight>1.6</weight>
  <rating>79</rating>
</courses>
<module>
  <name>Software Development</name>
  <startDate>2016-03-23</startDate>
  <endDate>2016-09-23</endDate>
</module>
</student>
<student>
...
</student>
</students>

```

Das Format wurde den gegebenen Umständen angepasst, damit die Transformierung so einfach wie möglich durchgeführt werden kann. Dies erleichterte auch die Anpassung des Stylesheet Files.

Die Stylesheets selbst wurden aus dem Rahmenprogramm übernommen. Nach dem Erstellen des Leistungsnachweis-Template wurden die Stylesheets auf beide Templates abgestimmt. So wird nun je ein Template für das Zertifikat und den Leistungsnachweis benötigt. Die Templates werden dann mit einem Stylesheet transformiert in ein XHTML. Daraus wird dann mit einem XSL-FO Stylesheet das XHTML in ein FO Dokument transformiert. Danach wird mit dem FOP ein PDF erzeugt.

Dies ermöglicht es dem Anwender, es sehr einfach zu erweitern, falls dies einmal nötig oder gewünscht wird.

6 Implementation

6.1 Zugriffsbeschränkung

Der Zugriff auf die Applikation wird mit einem einfachen Login beschränkt. Die Userdaten fürs Login sind in der Datenbank gespeichert, welche ebenfalls mit einem Login gesichert sind. Der Zugriff auf die Datenbank sollte auf ein Minimum beschränkt werden. So Vielen wie nötig Zugriff geben. Die User-Accounts der DB sollten regelmässig überprüft und alte User Accounts gelöscht werden.

6.2 Notenberechnung

Die Berechnung der Noten erfolgt direkt in der Transformation der XML Rohdaten ins XHTML Format. Die Formel dazu lautet: Addieren aller Noten mal seines Gewichts, geteilt durch die Summe aller Gewichte:

```
<tr>
    <td id="bold">Gesamt Erfolg:</td>
    <td></td>
    <td id="bold"><xsl:value-of select="format-number(sum(for $x in $courses return sum($x/rating * $x/weight)) div sum($courses/weight), '#.0')"/></td>
</tr>
<tr>
    <td id="bold">ECTS-Note:</td>
    <td></td>
    <td id="bold"><xsl:choose>
        <xsl:when test="sum(for $x in $courses return sum($x/rating * $x/weight)) div sum($courses/weight) > 89">A</xsl:when>
        <xsl:when test="sum(for $x in $courses return sum($x/rating * $x/weight)) div sum($courses/weight) > 79">B</xsl:when>
        <xsl:when test="sum(for $x in $courses return sum($x/rating * $x/weight)) div sum($courses/weight) > 69">C</xsl:when>
        <xsl:when test="sum(for $x in $courses return sum($x/rating * $x/weight)) div sum($courses/weight) > 59">D</xsl:when>
        <xsl:when test="sum(for $x in $courses return sum($x/rating * $x/weight)) div sum($courses/weight) > 49">E</xsl:when>
        <xsl:when test="sum(for $x in $courses return sum($x/rating * $x/weight)) div sum($courses/weight) <= 50">F</xsl:when>
    </xsl:choose>
    </td>
</tr>
```

6.3 Dokumenterzeugung

Die Dokumentenerzeugung wurde vom Rahmenprogramm abgeleitet und auf unsere Bedürfnisse angepasst. Pro Dokument (Certificate, Leistungsnachweis) gibt es ein Template. Daraus wird jeweils mit einem Stylesheet die Transformation der Templates in ein XHTML und die Transformation des erzeugten XHTMLs in ein XSL-FO Dokument gemacht. Falls die Erzeugung noch auf das Diplom erweitert werden müsste, muss nur noch ein weiteres Template erstellt werden und eine weitere Methode im CasAssistantViewController eingefügt werden.

7 Erreichte Ziele und gemachte Erfahrungen

Wir konnten im Verlauf der Projektarbeit alle gestellten Anforderungen umsetzen. Die Applikation kann nicht ohne Login benützt werden. Die View jedes Benutzers wird seiner Rolle angepasst, gemäss den Angaben in der Datenbank. Jeder Dozent kann die Noten seiner Kurse und Studenten einsehen, erfassen und bearbeiten. Jeder CAS-Verantwortliche kann alle Noten der Kurse und Studenten einsehen, erfassen und bearbeiten. Jeder Assistent kann die Noten der CAS einsehen und wenn die Noten komplett sind, einen Leistungsnachweis ausdrucken. Für alle Student, die einen Durchschnitt über 50% haben, wird auch gleich ein Zertifikat ausgedruckt. Jeder Student kann seine Noten einsehen. Die Applikation wurde so gut wie möglich nach ergonomischen Richtlinien erstellt.

Die Applikation wurde so aufgebaut, dass sie leicht erweitert werden kann. Die Daten sind alle durch ein Login geschützt. Schreibvorgänge in die Datenbank werden durch ein Optimistic Locking gesichert.

Die Applikation wurde mit einigen JUnit Tests getestet, jedoch konnte sehr wenig getestet werden, da sehr vieles von der Datenbank abhängig ist und somit keine Unitests gemacht werden können.

Der gesamte Code wurde - wo sinnvoll - mit Java Docs Kommentaren versehen und mit Checkstyle geprüft.

Während der Entwicklung haben wir auch Situationen erlebt, in denen wir eine Funktion fertiggestellt hatten und danach bemerkten, dass es nicht so funktioniert wie es sollte. So mussten wir mehrmals den Code anpassen, welchen wir eigentlich fertiggestellt hatten. Das Generieren der Dokumente wurde einmal erstellt und danach komplett umgeschrieben, damit die Erweiterbarkeit gewährleistet ist.