CAS SD: Projekt EQualS

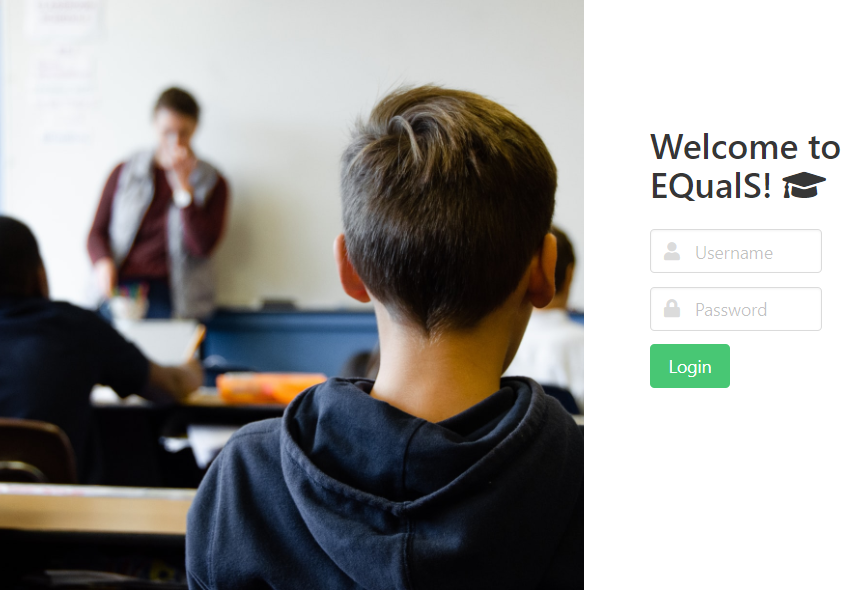
Datum: 9. März 2021

Autoren: Raphael Gerber, Christophe Leupi, Sabina Löffel, Igor Stojanovic

# Projekt EQualS

# Gruppe 1

# Raphael Gerber, Christophe Leupi, Igor Stojanovic, Sabina Löffel



[1. Use Cases 2](#_Toc66796044)

[1.1. Use Case Diagramm 2](#_Toc66796045)

[2. Datenmodell 3](#_Toc66796046)

[2.1. ER-Modell 3](#_Toc66796047)

[2.2. Datenbank-Schema 4](#_Toc66796048)

[2.3. Datenbank-Schema: Konsolidierung 5](#_Toc66796049)

[3. Graphische Benutzeroberfläche 6](#_Toc66796050)

[3.1. Ergonomie-Überlegungen 6](#_Toc66796051)

[3.2. GUI-Prototyp 6](#_Toc66796052)

[4. Gesamtdesign 8](#_Toc66796053)

[4.1. Design-Überlegungen 8](#_Toc66796054)

[4.2. Architektur 8](#_Toc66796055)

[4.3. Rest-Schnittstelle 9](#_Toc66796056)

[5. Implementation 10](#_Toc66796057)

[5.1. Zugriffskontrolle 10](#_Toc66796058)

[5.2. Notenberechnung 13](#_Toc66796059)

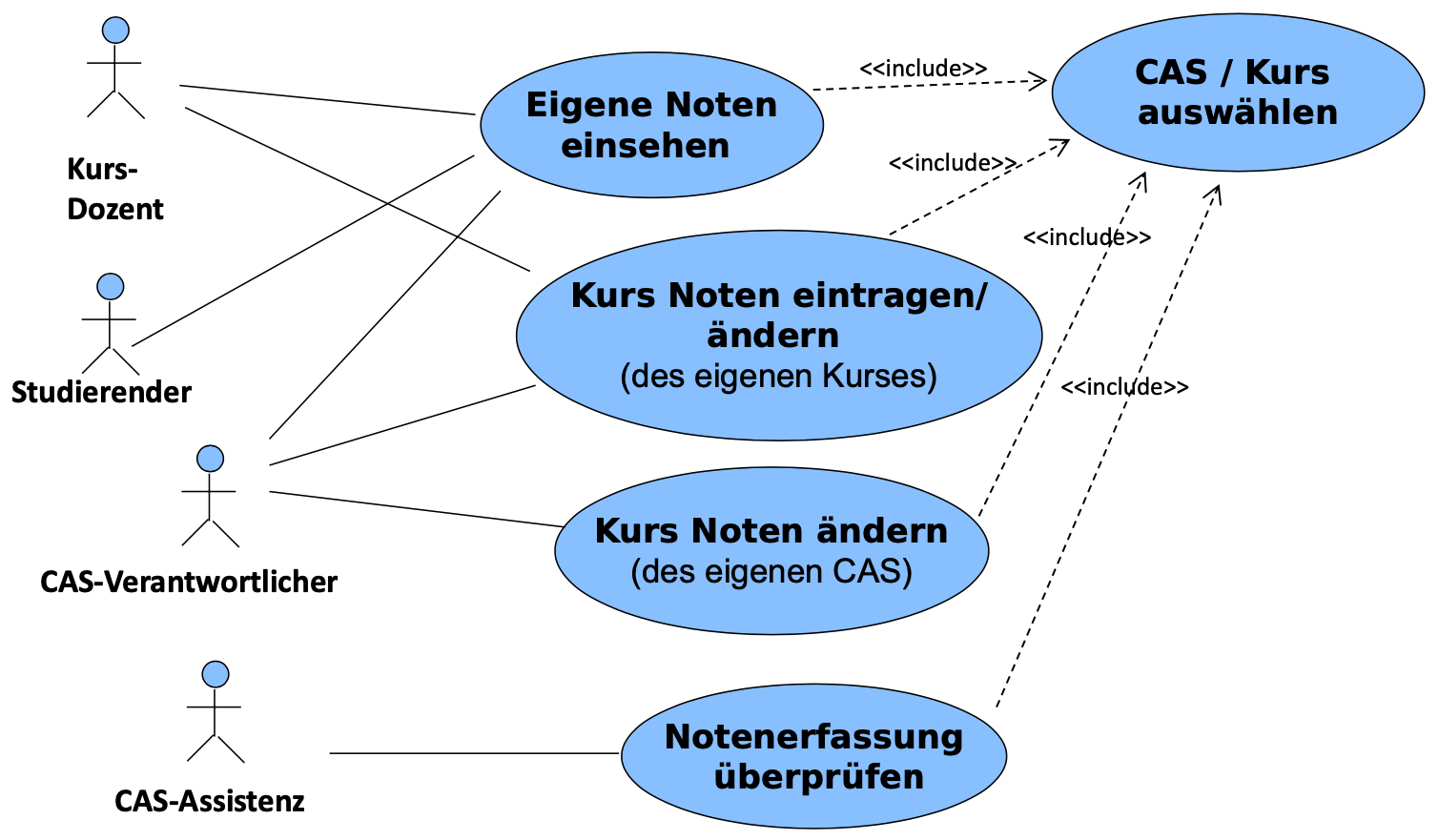
[6. Erreichte Ziele und gemachte Erfahrungen 14](#_Toc66796060)

[7. Glossar 15](#_Toc66796061)

# Use Cases

## Use Case Diagramm

Die Projektgruppe erarbeitete die Projektarbeit auf der Grundlage der Aufgabenstellung:

**

**Kurs-Dozenten (Teacher)**

Kursdozenten müssen sich in der Applikation einloggen, einen Kurs auswählen und sind dann berechtigt Noten in den Kursen, die sie unterrichten, zu erfassen, einzusehen und zu modifizieren. Auf anderen Kursen haben sie keinen Zugriff.

**Studierende**

Studierende müssen sich in der Applikation einloggen, ein CAS selektieren und die eigenen Kursnoten einsehen.

**CAS-Verantwortliche**

CAS-Verantwortliche müssen sich in der Applikation einloggen, ein CAS, welches von ihnen verantwortet wird, selektieren und alle Kursnoten aller Studierenden einsehen und ändern. Die CAS-Verantwortlichen können nur Noten der Kurse erfassen, welche sie selbst unterrichten (analog Kurs-Dozenten). Pro CAS gibt es nur eine verantwortliche Person.

**CAS-Assistenz**

Die CAS-Assistenz kann in allen CAS, welche von ihr assistiert werden, die Noten einsehen und die Dozenten auf fehlende Noten aufmerksam machen. Sie verfügen nur über Lese- und über keine Schreibrechte. Pro CAS gibt es nur eine Assistenz.

**Generell:**

Es gilt zu beachten, dass eine Person mehrere Rollen haben kann in unterschiedlichen CAS. So ist es möglich, dass ein Dozent einen Kurs unterrichtet, in einem anderen CAS als Student tätig sein. Auch ist es so möglich, dass eine CAS-Assistenz in einem anderen CAS als Studierende/r eingeschrieben ist.

# Datenmodell

## ER-Modell

Unser ER-Modell fasst die Rollen Student / Dozierender / CAS-Verantwortlicher / CAS-Administration in eine Rolle, namentlich Person, zusammen. Die Möglichkeiten der Rollen sind im Kapitel 1 «Use Cases» zusammengefasst. Dieses Modell bietet die Flexibilität, dass keine Mehrfacheinträge von gleichen Personen in verschiedenen Rollen in der Datenbank vorhanden sind, wenn z.B. ein CAS-Verantwortlicher auch gleichzeitig ein Dozent oder ein Dozent gleichzeitig ein Student ist.

Im ER-Modell ist dadurch eine zirkuläre Abhängigkeit entstanden, welche auf den ersten Blick problematisch wirkt, jedoch bei einem Projekt dieser Grösse durchaus erfolgreich umgesetzt werden kann. Daraus ergibt sich gleichzeitig auch eine Schwäche im Modell, welche es ermöglicht, einen Studierenden als CAS Verantwortliche/n einzutragen. Die Projektgruppe ist sich dieser Schwäche bewusst, ist jedoch überzeugt, dass es sich lohnt ein einfaches Modell für die Umsetzung bereitzustellen (KISS – Keep it Simple and Short).

Ein CAS-Verantwortliche/r kann mehrere CAS leiten, wobei ein CAS nur eine/n Verantwortliche/n haben kann (analog CAS-Assistenz). Ein CAS besteht aus mehreren Kursen und ein Kurs kann genau zu einem CAS gehören. Dies ist diskutierbar und könnte vorausschauend anders aufgebaut werden, die Projektgruppe einigte sich jedoch im Rahmen des Projektes dies so beizubehalten. Studierende können sich in mehrere CAS einschreiben, wobei auch mehrere Studierende in einem CAS-Kurs eingeschrieben sein können. Die CAS-Note ist ein kalkulierter Eintrag, der die Gewichtung der Kurse (quantifier) berücksichtigt und erst berechnet wird, wenn in allen zu einem CAS zugehörigen Kursen mit Noten ausgestattet wurden.

**

## Datenbank-Schema



Das Datenbank-Schema basiert auf dem ER-Modell, wurde jedoch um weitere Attribute bei den Entitäten und Relationen ergänzt. Die Projektgruppe verzichtete aufgrund Komplexitäts-minderungen bewusst auf die Aufnahme von boolean-Spalten in der Personen-Entität (z.B. isStudent, isTeacher, isCASResponsible, isAssistant). Die Constraints im Modell verunmöglichen es, z.B. Teacher in der Tabelle Course\_Teacher zu erfassen, ohne dass es die entsprechende ID in der Personentabelle gibt. Hier folgen noch einige Testqueries für das erarbeitete Modell:

**Beispiel 1:**

SELECT p.lastName, cs.grade FROM Person p  
Inner join Course\_Student cs on cs.studentId = p.id  
Inner join CAS\_Student cass on cass.StudentId = p.id  
Inner join CAS cas on cass.CASId = cas.id  
WHERE cas.id = 1 AND p.id = 1;

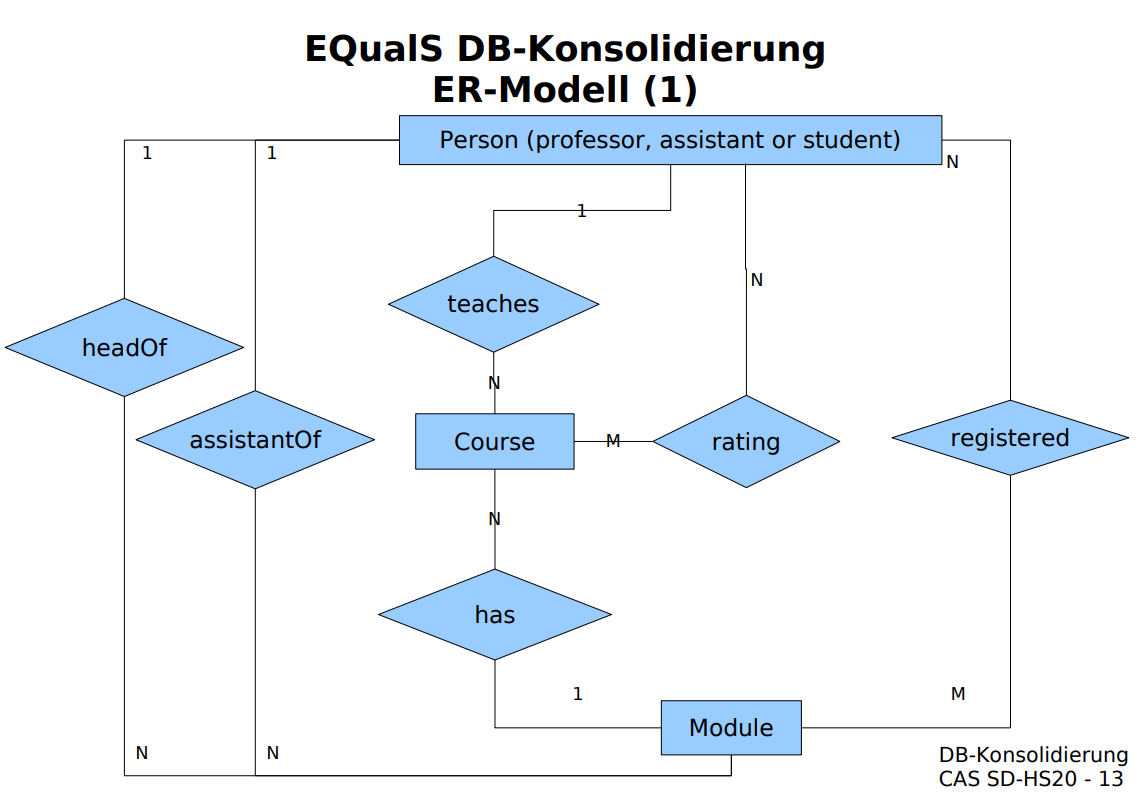
*Nachname und Noten der Besuchten Kurse einer Person mit der CAS ID 1 und Personen ID 1*

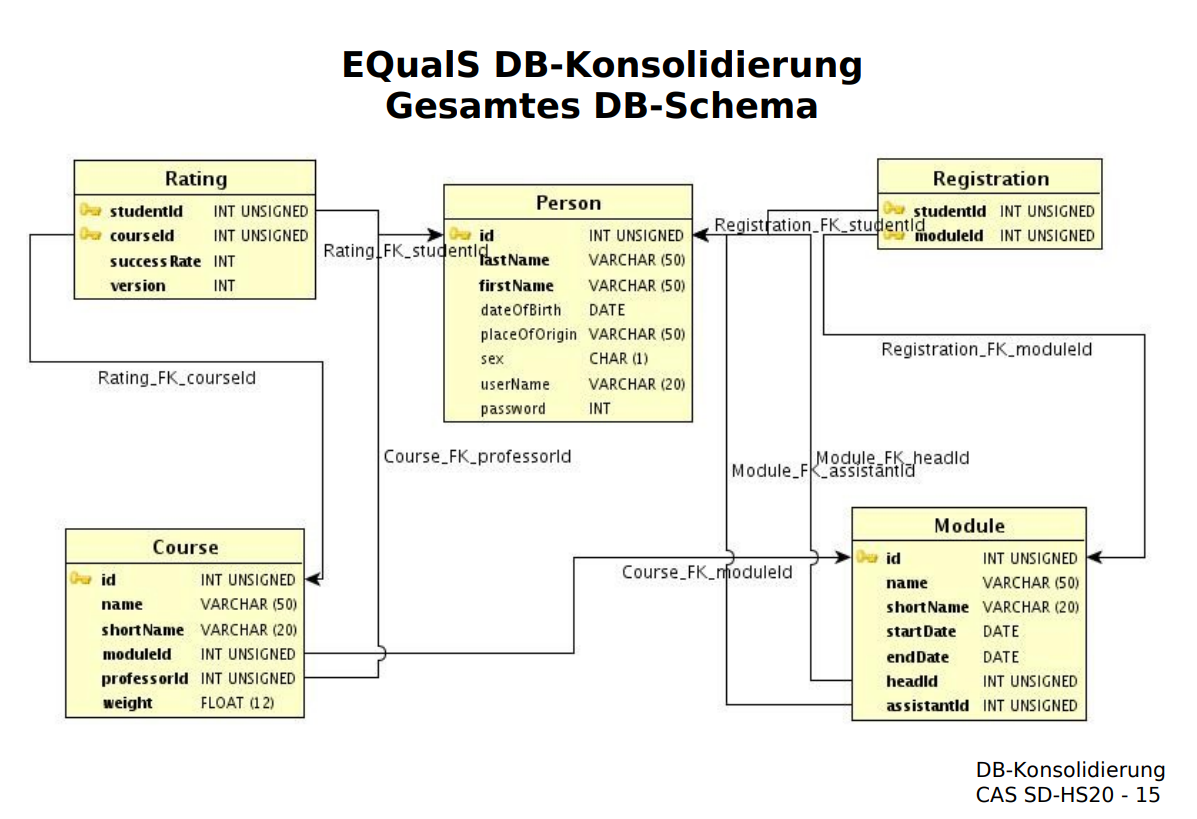
**Beispiel 2:**

SELECT p.lastName FROM Person p  
Inner join Course\_Teacher teacher on p.id = teacher.teacherId  
Inner join Course c on teacher.courseId = c.id  
Inner join CAS cas on c.CASId = cas.id  
WHERE cas.id = 22;

*Alle Personen-Nachnamen, welche im CAS mit der ID 22 unterrichten (Teacher).*

## Datenbank-Schema: Konsolidierung





# Graphische Benutzeroberfläche

## Ergonomie-Überlegungen

Die Login Maske sieht für alle verschiedenen Rollen identisch aus. Der Benutzer kann sich mit dem Usernamen und Passwort «stud» einloggen. Nach erfolgtem Login sind die Module in einer Übersicht als Kacheln ersichtlich. Dabei werden jeweils nur die Module angezeigt, worauf der Benutzer Berechtigungen hat. Module, die zeitlich bereits in der Vergangenheit liegen, werden ausgegraut dargestellt.

* + 1. **Head / Professor**

Benutzer mit den Rollen Head oder Professor können Noten eintragen und verändern. Dem Benutzer Head werden alle Kurse des Moduls angezeigt. Dem Benutzer Professor werden nur diejenigen Kurse angezeigt, bei denen er Kursleiter ist.

* + 1. **Assistenz**

Die Rolle Assistenz soll möglichst schnell eine Übersicht über Module mit fehlenden Noten erhalten. Ausserdem können Dozenten, welche noch nicht alle Noten eingetragen haben, per Mail benachrichtigt werden.

* + 1. **Student**

Der Student kann nur seine Noten von seinen besuchten Kursen einsehen.

Für die Rollen Head, Professor und Assistenz werden Module, wo Noten für gewisse Kurse noch nicht eingetragen sind, mit dem Label «Missing grades» dargestellt. Ausserdem können die Module nach Semester und «Missing grades» gefiltert werden.

**Save Button auf der Module Overview**

Der Save Button wird initial ausgegraut angezeigt (nach erfolgtem Login und Navigation auf die Übersicht). Wenn Werte aus der Liste verändert werden, wird der Button aktiviert und kann selektiert werden. Die Werte werden erst gespeichert, nach dem Abschicken vom Save Button. Bei der Eingabe der Werte gibt es Formatprüfungen auf die zulässigen Werte, die dem Benutzer als Orientierung dienen. Wenn unzulässige Werte eingegeben werden, kann der Request nicht abgeschickt werden und es wird ein Fehler angezeigt. Nach erfolgter Speicherung wird eine Bestätigungsmeldung ausgegeben.

## GUI-Prototyp

|  |  |
| --- | --- |
| Login Mock |  |
| Module Overview Mock |  |
| Courses Professor View Mock |  |
| Course Assistant View Mock |  |
| Course Rating Overview Mock |  |
| Notify Mock |  |
| Student View Mock |  |

# Gesamtdesign

## Design-Überlegungen

Wir haben uns für eine Architektur mit Business, Controller und Repository-Klassen entschieden. Die Repository regelt die Zugriffe auf die Datenbank, die Kontroller-Klassen sind zuständig für die REST-Calls und die Business-Klassen beinhalten die Geschäftslogik.

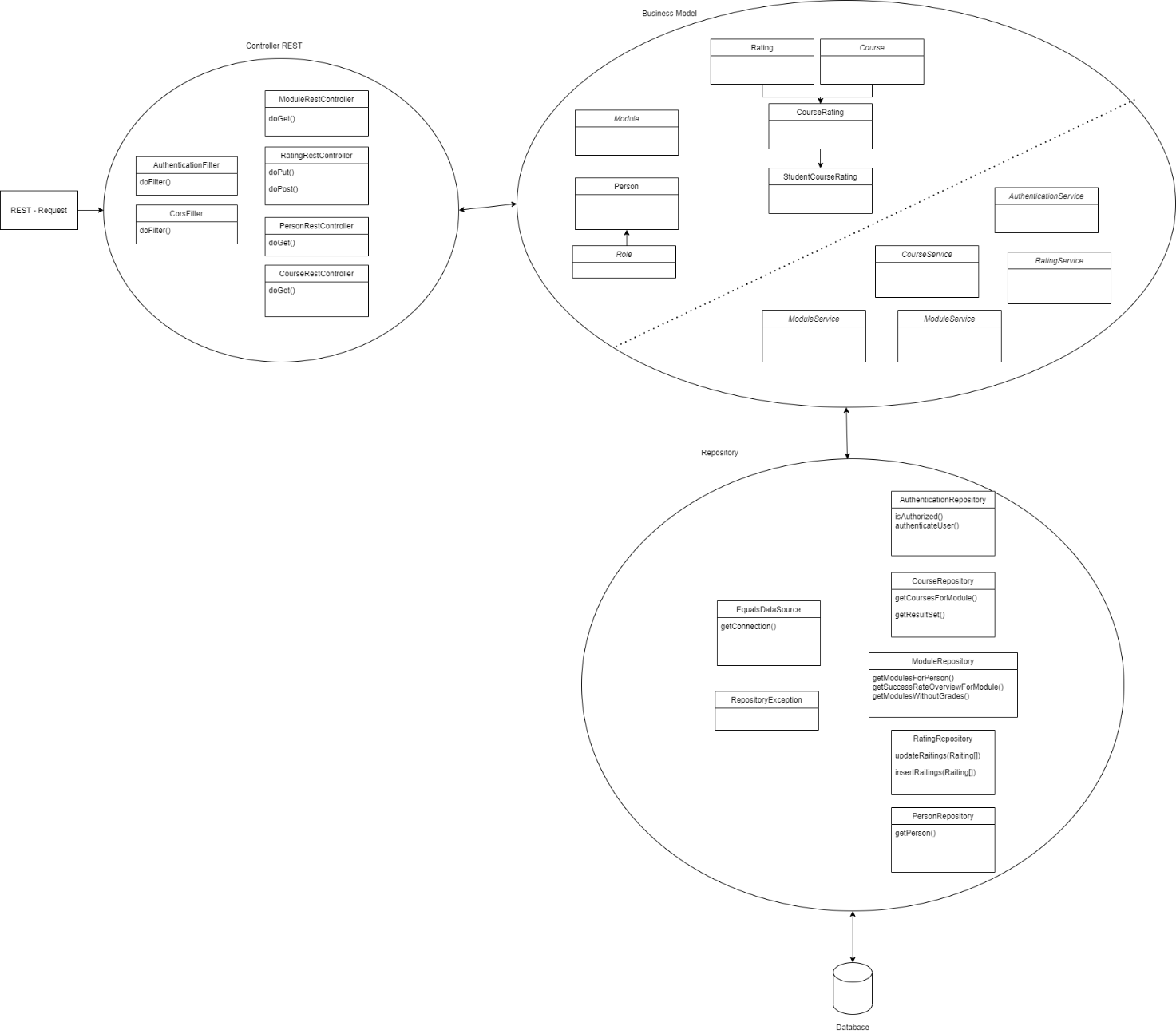
Die Controller-Klassen greifen über den Service-Layer auf die Business-Klassen zu, damit die Demeter-Regeln eingehalten werden.

Wir haben uns entschieden für alle Rollen einen einheitlichen Einstiegspunkt (über die Module) anzubieten. Dadurch ist die Applikation sehr einheitlich und es ist auf einen Blick ersichtlich, wie der Zustand der Module / Kurse ist durch die farbliche Unterstützung.

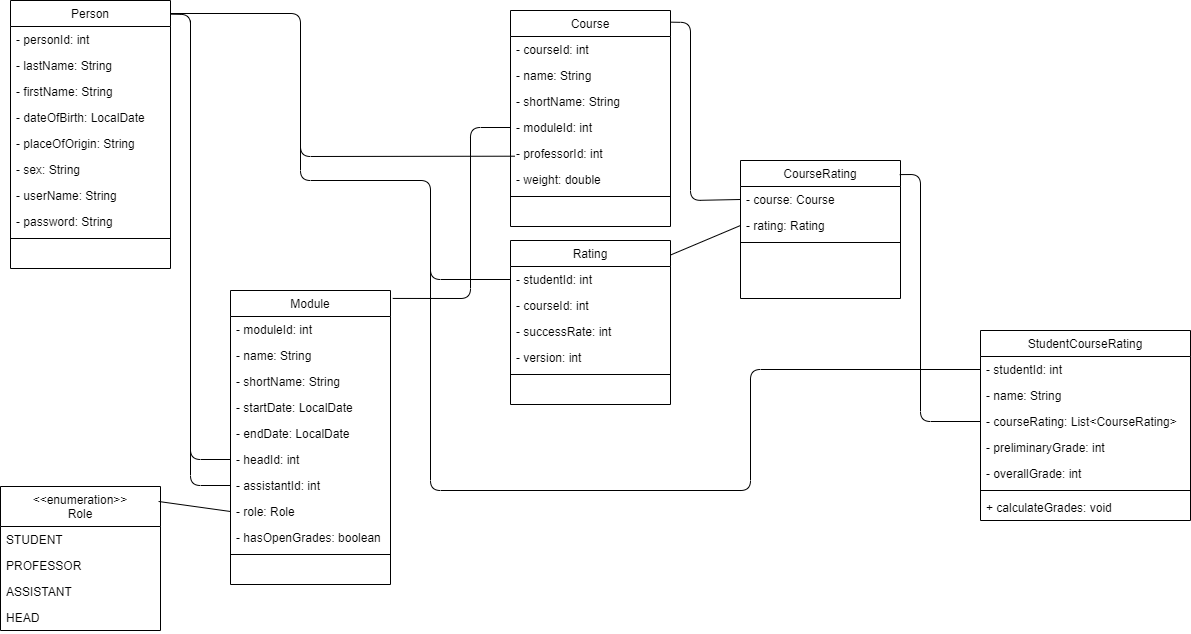
## Architektur

*Z.B. Grob-Klassendiagramm mit Erklärungen, …*

*DRAFT*



*DRAFT*



## Rest-Schnittstelle

Das REST API ist über http://localhost:8080/equals\_war/ erreichbar und beschreibt mit der jeweiligen Authentifizierung die entsprechenden Endpoints von unserem Backend. Die Open API Specification ist über die URL <http://localhost:8080/equals_war/swagger> verfügbar.

Folgende Pfade haben stehen via REST API zur Verfügung:

persons

* *GET /persons*
* *GET /persons/{id}*

modules

* *GET /modules*
* *GET /modules/overall/{id}*

courses

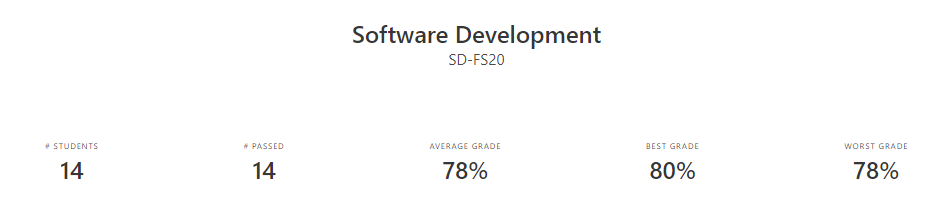
* *GET /courses/{moduleId}*

ratings

* *POST /ratings*
* *PUT /ratings*

# Implementation

Die Module Overview zeigt für alle Rollen ausser für den Student Statistik an über das CAS. Man sieht die Anzahl Studenten, die Anzahl erfolgreichen Studenten, die Durchschnitts-Note sowie die beste und schlechteste Note.



Bei allen Rollen ist ein Button «Back to Modules» vorhanden. Zusätzlich ist für die Rollen Head und Professor ist der Button Save vorhanden auf der Module Overview. Bei der Rolle Assistant ist ein ausgegrauter Print-Button vorhanden. Damit könnte später eine Druck-Funktion einfach implementiert werden.

## Zugriffskontrolle

Damit die Daten je unterschiedliche Benutzer geschützt sind vor unerlaubten Zugriffen haben wir verschiedene Views erstellt. Auf jedem Modul ist die Rolle des eingeloggten Benutzers angezeigt mit einem Tag.

* + 1. **Head**

Wenn für ein CAS Noten nicht eingegeben worden sind, sind diese Module mit dem Tag «Missing grades» ergänzt. Auf der Module Overview Seite können die Filter «Missing grades» und Semester angewendet werden.

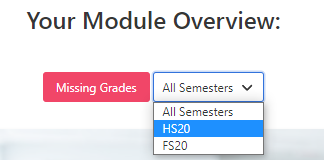
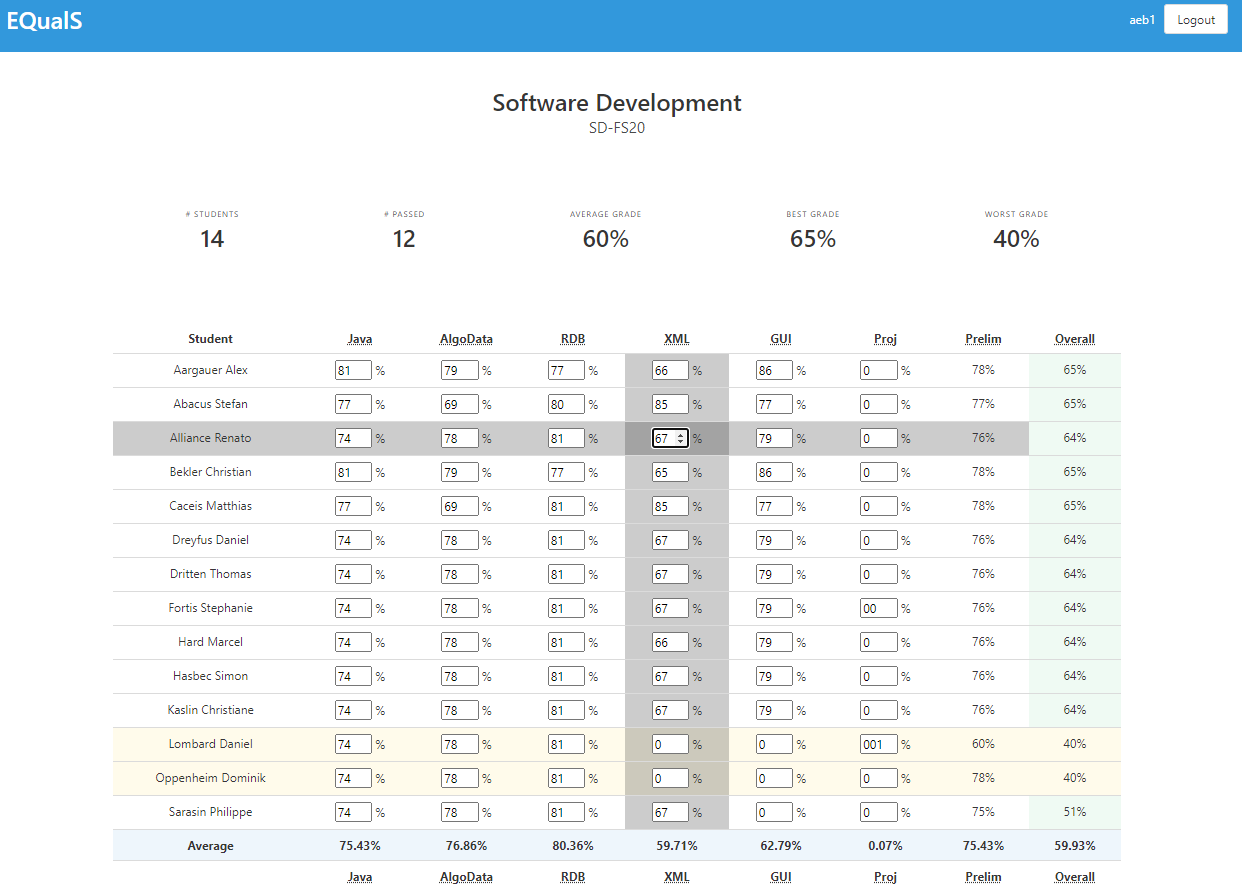


Abbildung 1: Filter nach Missing Grades und / oder Semester

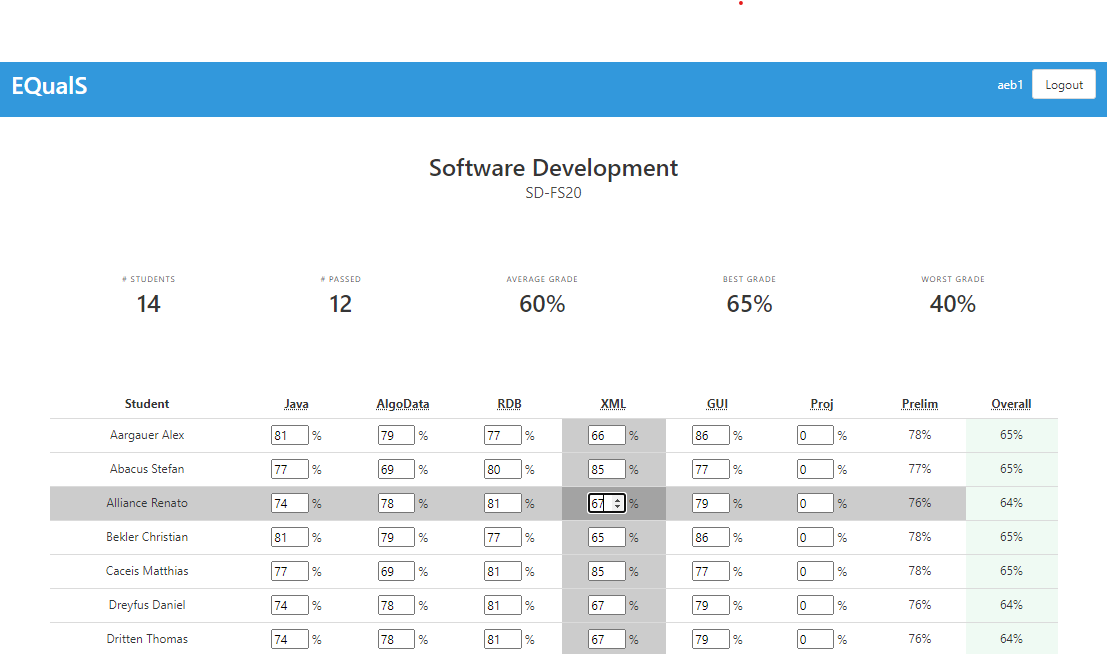


Abbildung 2 : Darstellung Tag Missing grades & Tag Head

Wenn das CAS bestanden ist, wird die Overall Spalte grün hinterlegt. Wenn das CAS nicht bestanden ist, wird Overall und die ganze Studenten-Zeile gelb hinterlegt dargestellt. Pro Kurs wird die Durchschnitts-Note der Studenten in der Zeile Average berechnet und angezeigt. Die Zeile wird blau hinterlegt dargestellt. Zur einfachen Lesbarkeit sind die Namen der jeweiligen Kurse zuunterst tabellarisch noch einmal aufgeführt.



Zur Eingabeunterstützung wird die aktuell selektierte Zeile und Spalte grau hinterlegt.



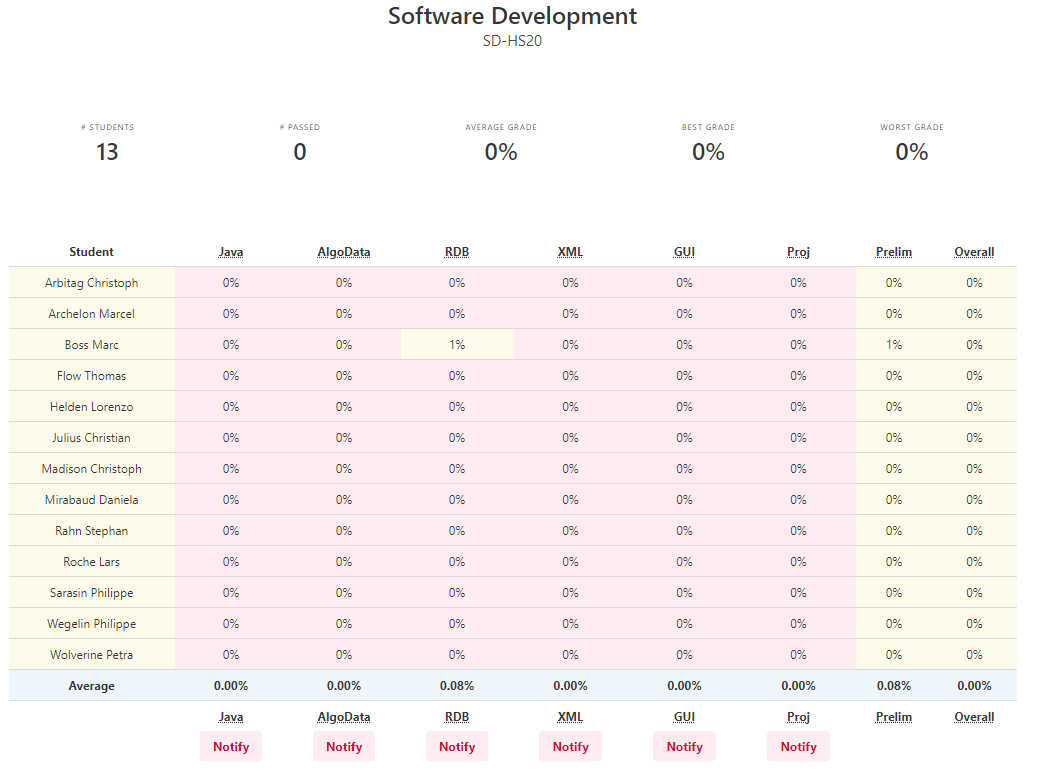
* + 1. **Professor**

Die Funktionen decken sich mit der HEAD View, ausser das nur diejenigen Kurse angezeigt werden, wo der Benutzer auch die Rolle Professor einnimmt.

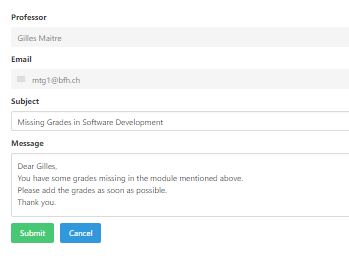
* + 1. **Assistant**

Bei der Module Overview werden fehlende Noten rot hinterlegt angezeigt. Die Noten können von der Rolle Assistant nicht verändert werden.

Sobald eine Note für einen Kurs fehlt, kann mit der Notify – Funktion eine E-Mail an den Professor erstellt werden mit einem Default Text.

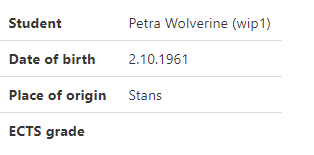


Die Notify-Meldung löst über den lokal installierten E-Mail Client den Versand von einem Reminder aus.



* + 1. **Student**

Dem Studenten werden die besuchten Module angezeigt und die darin enthaltenen Kurse sowie die Gewichtung der Noten pro Kurs. Zusätzlich werden weitere Angaben zum Studenten angezeigt.



## Notenberechnung

Die Notenberechnung wird aktuell im Frontend und im Backend durchgeführt.

Der Preliminary Grade rechnet den Zwischenstand der eingegebenen Noten der Kurse und berücksichtigt die fehlenden Noten nicht. Hingegen der Overall-Grade rechnet den Durchschnitt aller Grades der Kurse. Die Noten werden alle gerundet.

# Erreichte Ziele und gemachte Erfahrungen

Wir haben während der gesamten Arbeit stets den Ansatz «make it work, make it right, make it fast.» angewendet. Wir konnten unser erlerntes Wissen im Bereich JavaScript in die Praxis umsetzen und kamen gut voran.

Wir haben die Unittests zusammen erarbeitet und dabei Mockito eingesetzt, um die Unittests zu mocken.

Die Dokumentation unserer REST API über Swagger ging relativ ring und wir haben unser Wissen erweitert. Dabei konnten wir noch abfangen, dass bei einer fehlenden oder falschen ID im /modules und /persons eine entsprechende Error-Meldung geworfen wird.

Durch die Arbeiten mit Swagger ist ebenfalls aufgekommen, dass ein Student ebenfalls Ratings an unser Backend senden kann. Dies mussten wir nachträglich noch korrigieren.

Bei Git konnten wir alle unsere Kenntnisse erweitern und vertiefen und haben durch die praktische Arbeit einige Fortschritte erzielt.

Zu Beginn des Projekts waren wir sehr JavaScript fokussiert und haben uns somit erst später mit dem Design der REST API befasst. Dadurch ist das Design der REST Schnittstelle verbesserungswürdig.

# Glossar

Wir haben folgende Begriffsdefinitionen angewendet:

|  |  |
| --- | --- |
| **Module** | CAS |
| **Course** | Kurs |
| **Grade** | Note pro CAS / Note pro Modul – wird verwendet für die Gesamt- und Zwischen-Berechnung |
| **Rating** | Quote, Erfolgsquote / Einzelnote pro Kurs |
| **Preliminary** | Zwischenerfolg |
| **Overall** | Gesamterfolgsquote/note |