CAS SD: Projekt EQualS

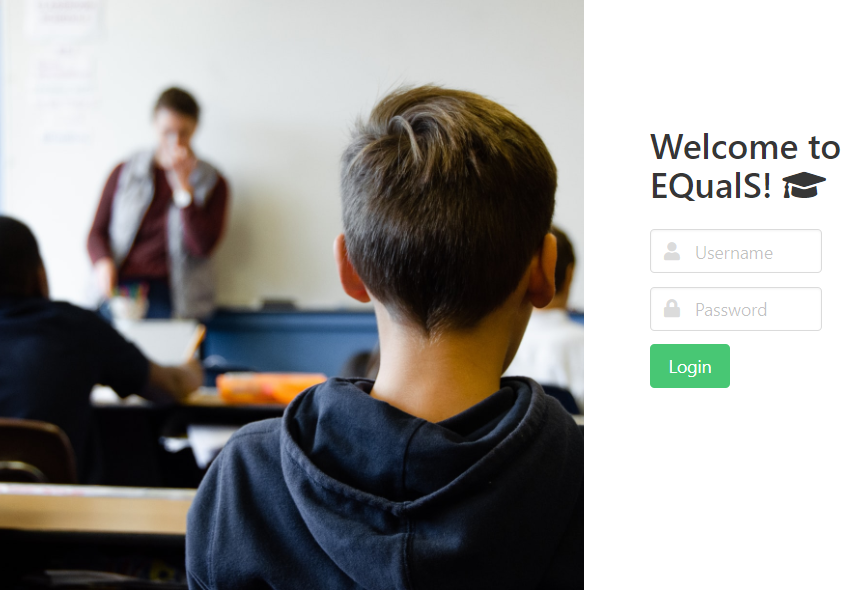
Datum: 21. März 2021

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

# Projekt EQualS

# Gruppe 1

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



Inhaltsverzeichnis

[1. Use Cases 3](#_Toc67231900)

[1.1. Use Case Diagramm 3](#_Toc67231901)

[2. Datenmodell 4](#_Toc67231902)

[2.1. ER-Modell 4](#_Toc67231903)

[2.2. Datenbank-Schema 5](#_Toc67231904)

[2.3. Datenbank-Schema: Konsolidierung 6](#_Toc67231905)

[3. Graphische Benutzeroberfläche 7](#_Toc67231906)

[3.1. Ergonomie-Überlegungen 7](#_Toc67231907)

[3.2. GUI-Prototyp 7](#_Toc67231908)

[3.3. App Walk-Through 9](#_Toc67231909)

[4. Gesamtdesign 11](#_Toc67231910)

[4.1. Design-Überlegungen 11](#_Toc67231911)

[4.2. Architektur 11](#_Toc67231912)

[4.2.1 Domänenmodel 11](#_Toc67231913)

[4.2.2 Drei-Schichten-Architektur 11](#_Toc67231914)

[4.3. Rest-Schnittstelle 13](#_Toc67231915)

[5. Implementation 14](#_Toc67231916)

[5.1. Zugriffskontrolle 14](#_Toc67231917)

[5.2. Notenberechnung 17](#_Toc67231918)

[6. Erreichte Ziele und gemachte Erfahrungen 19](#_Toc67231919)

[7. Glossar 20](#_Toc67231920)

# Use Cases

## Use Case Diagramm

Die Projektgruppe erarbeitete die Projektarbeit auf der Grundlage der Aufgabenstellung:

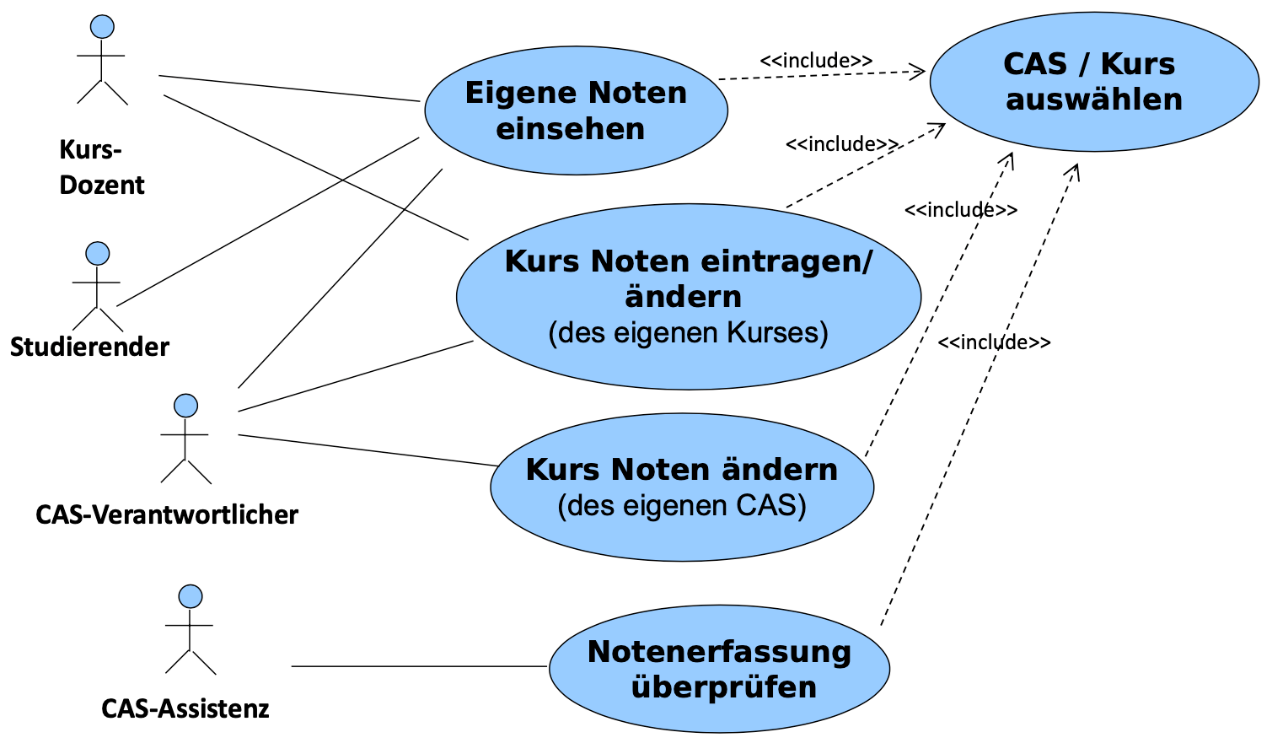
**

Abbildung 1: Use Case Diagramm

**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.

**

Abbildung 2: ER-Datenmodell

## Datenbank-Schema



Abbildung 3: DB 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;

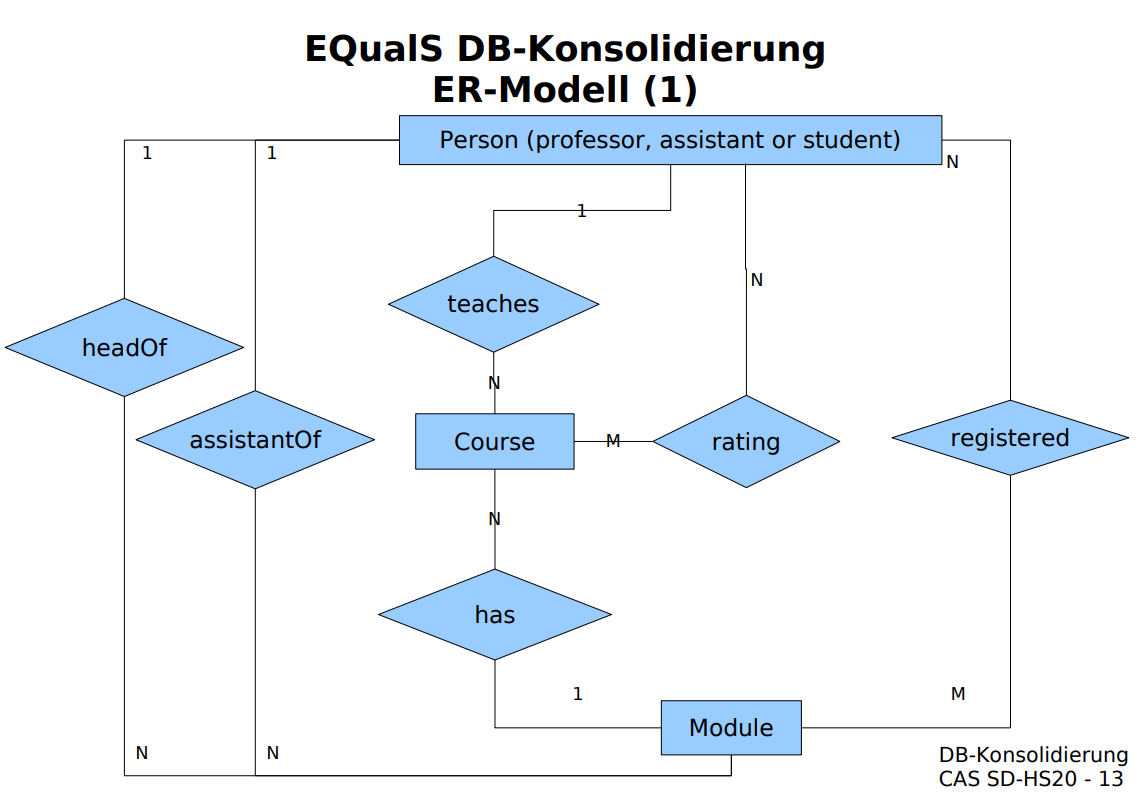
*Nachname und Noten der Besuchten Kurse einer Person mit der CAS ID 1 und Personen ID 1*

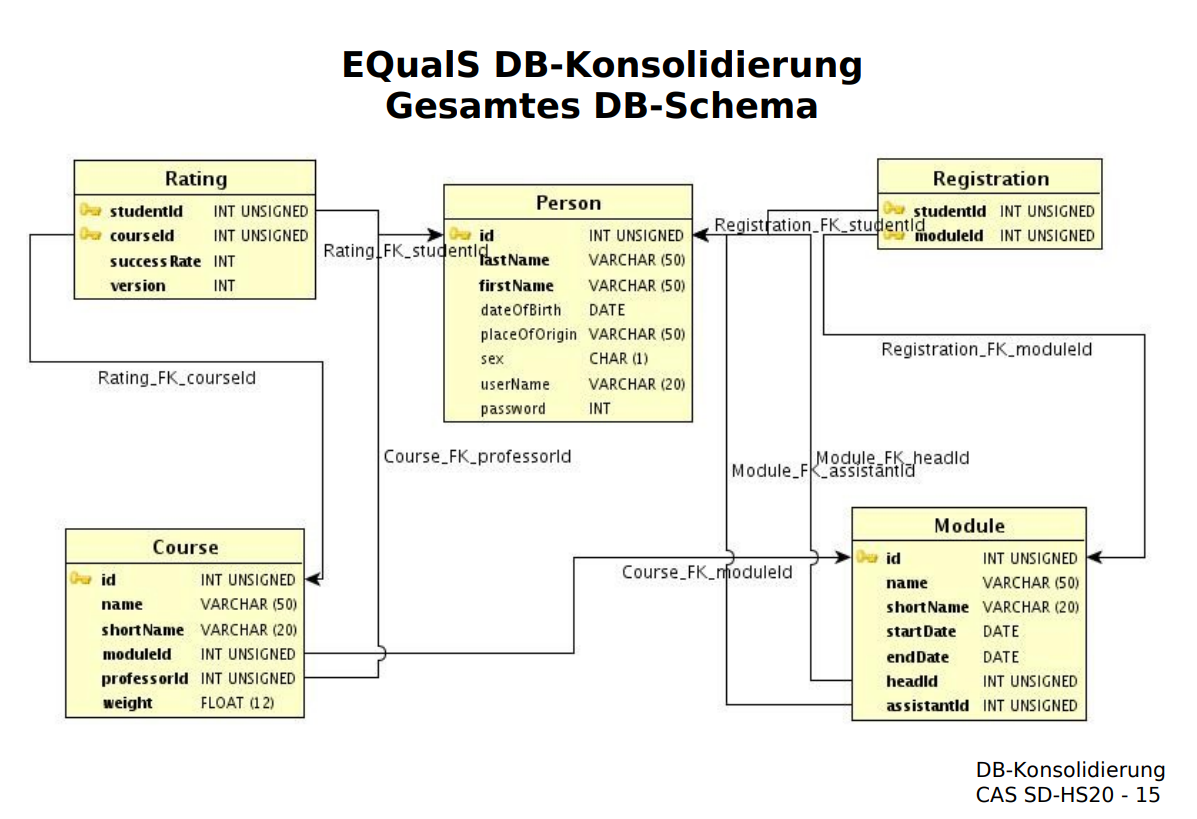
**Beispiel 2:**

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 (Assistant View) |  |
| Student View Mock |  |

## App Walk-Through

Als zentrale Anlaufstelle in der App dient die Overview-Komponente. Hier können alle User, unabhängig von der Rolle zwischen den einzelnen Modulen navigieren. Auf einem Blick stehen dem User alle Module zur Verfügung, wo eine Rolle wahrgenommen wird. Allen User-Rollen, ausser dem Student, steht eine Filterung zur Verfügung, da eine grosse Anzahl Modulen bei Studenten nicht erwartet wird. Für den CAS-Verantwortlichen, Professor und Assistent wird dieselbe Courses-Komponente aufgerufen, wobei sich die Ansicht je nach Rolle verändert wird (Anzahl Kurse, Inputfelder, etc.). Dem Assistenten wird zusätzlich ein Button angeboten, wo er direkt bei fehlenden Noten auf den Professor eines Kurses eine Nachricht (Email wird aufgerufen bei Submit) senden kann. Beim Student wird die Assessment-Komponente aufgerufen, welche eine unterschiedliche Darstellung aufzeigt, als bei den anderen Rollen. Das untenstehende Bild visualisiert den Page-Flow.

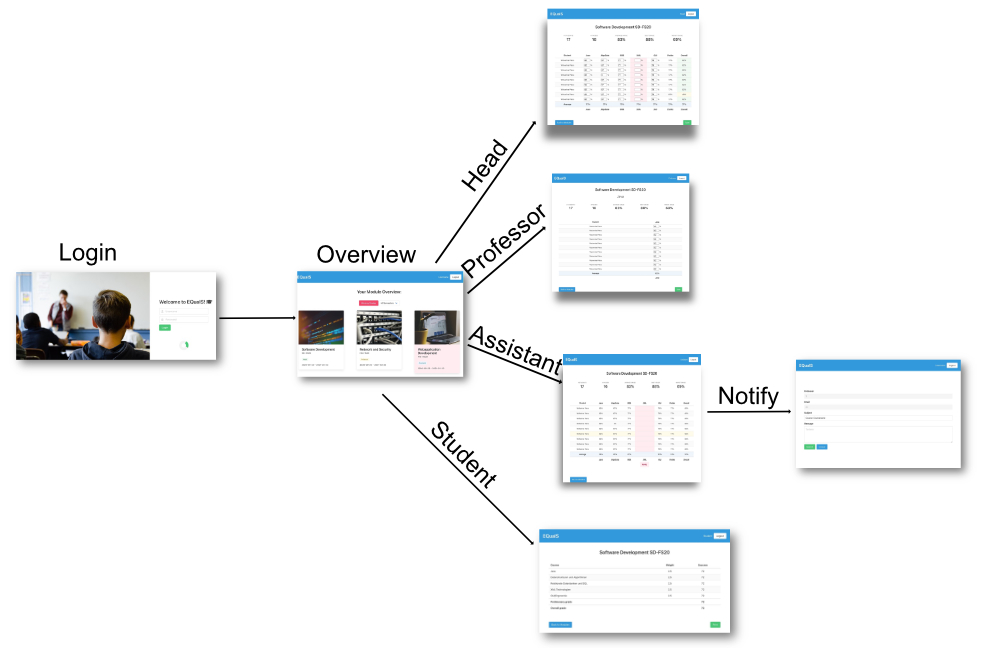


Abbildung 4: Page Flow

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.

# Gesamtdesign

## Design-Überlegungen

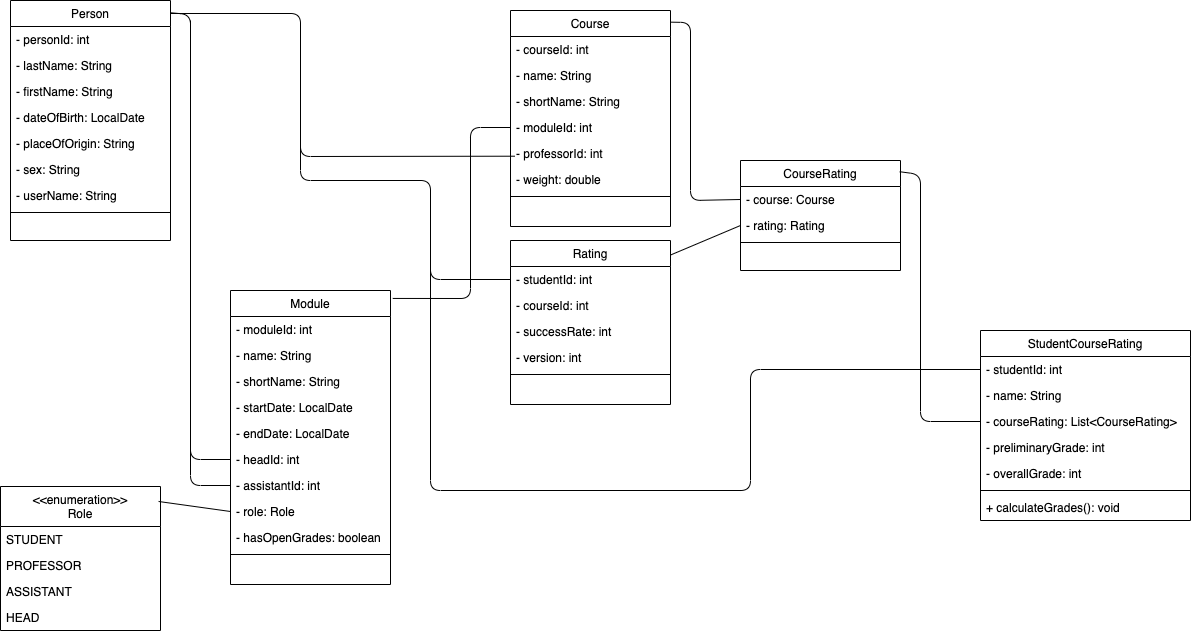
Wir haben uns für eine Drei-Schichten-Architektur mit Business/Service, Controller und Repository-Schichten entschieden. Die Repository regelt die Zugriffe auf die Datenbank, die Kontroller-Klassen sind zuständig für die Verarbeitung der REST-Calls und die Business-Klassen beinhalten die Geschäftslogik.

Die Controller-Klassen greifen über den Service-Layer auf die Business-Klassen und Repository zu, welche eine klare Trennung der Verantwortlichkeiten in der Drei-Schichten-Architektur ermöglicht.

## Architektur

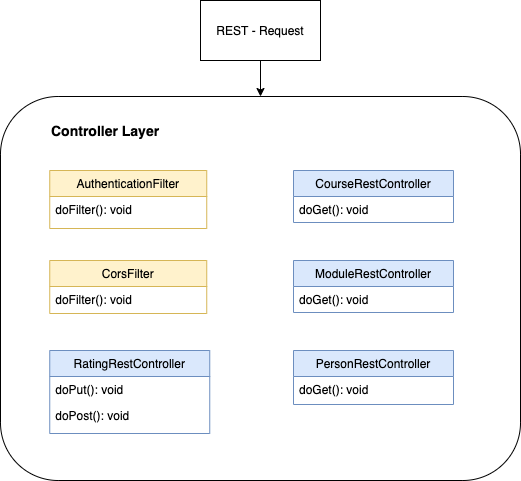
### 4.2.1 Domänenmodel

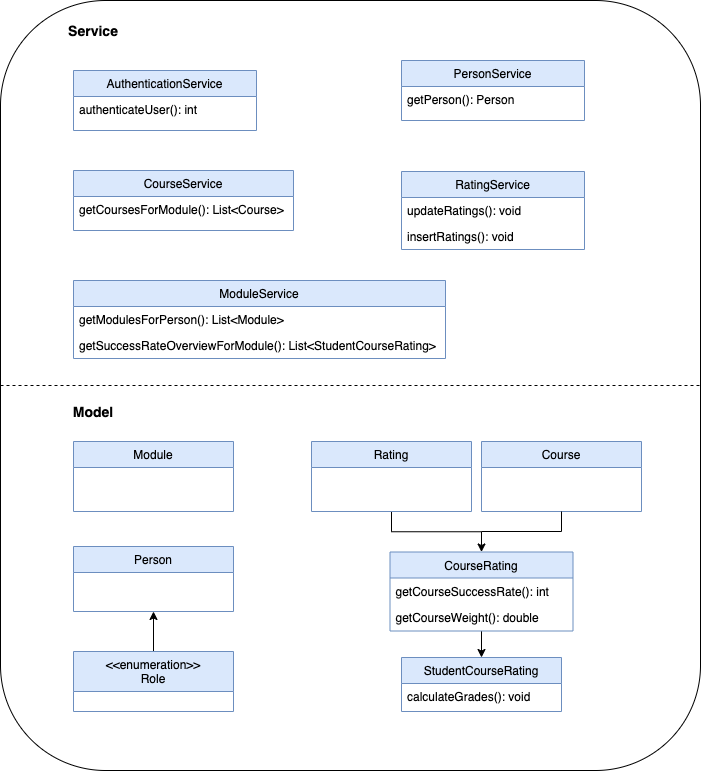
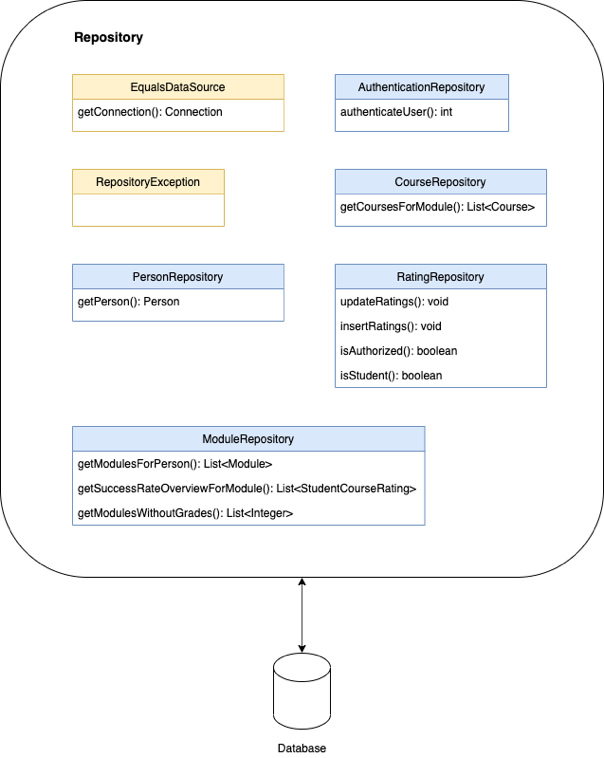
Das Domänenmodell ist stark an die Datenstruktur in der Datenbank angelehnt. Abweichend dazu, enthält die Personen Klasse das Attribut Password aus Sicherheitsgründen nicht. Zusätzlich zur Datenstruktur in der Datenbank wurde in der Module-Klasse die jeweilige Benutzer-Rolle und eine Markierung, welche nicht erfasste Noten aufzeigt, hinzugefügt. Ein Benutzer kann die Rolle eines Studenten, Dozenten, Assistenten oder CAS-Verantwortlichen einnehmen. Die Rolle kann jedoch je nach CAS variieren. Zur einfachen Ausgabe von Kursbewertungen wurde zusätzlich zur Course- und Rating-Klasse eine CourseRating-Klasse erstellt, welche schlussendlich ein Bestandteil des StudentCourseRatings sind. Das StudentCourseRating beinhaltet somit alle Informationen zu einem Modul, wie Kurse, Studenten und Noten. Dies reduziert die Anzahl benötigter API-Restaufrufe.

**

### 4.2.2 Drei-Schichten-Architektur

Ein Rest-Call dient als Einsteigpunkt in die Applikation. Vom Controller wird zuerst eine Authentifizierung mittels AuthenticationFilter durchgeführt. Dieser leitet die Anfrage an das entsprechenden Rest-Controller weiter. Dieser überprüft die Angaben und gibt die Anfrage an eine Service-Klasse weiter, welche die Repository-Schicht aufruft, welche die Model-Klassen befüllt. Ein Direktzugriff auf die Repository-Schicht von der Controller-Schicht aus wurde bewusst nicht implementiert. Die Repository-Schicht interagiert mit der Datenbank über JDBC.

**

**

## 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. Genauere Details der Implementierung können dort entnommen werden. Die einzelnen Aufrufe können lokal, direkt und interaktiv in Swagger abgesetzt werden.

Folgende Pfade haben stehen via REST API zur Verfügung:

persons

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

modules

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

courses

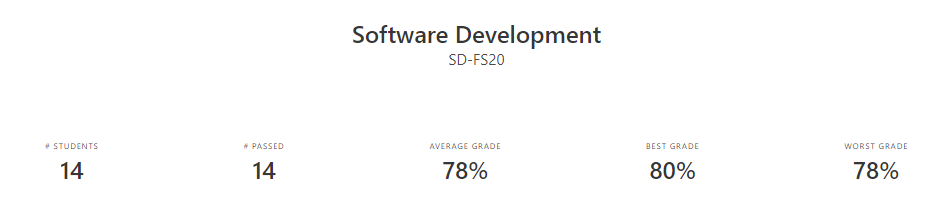
* *GET /courses/{moduleId}*

ratings

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

# Implementation

Die Module Overview zeigt für alle Rollen ausser für den Studenten Statistiken über das CAS an. Man sieht die Anzahl Studenten, die Anzahl erfolgreichen Studenten, die Durchschnitts-Note sowie die jeweils beste und schlechteste Note. Bei einem Dozenten wird die Berechnung auf Basis der gewichteten Kurse durchgeführt.



Bei allen Rollen ist ein Button «Back to Modules» vorhanden. Zusätzlich ist für die Rollen Head und Professor der Button Save vorhanden.

## Zugriffskontrolle

Die Zugriffskontrolle erfolgt beim Auslesen in der Repository-Schicht. Hier wird festgestellt, auf welche Module der Benutzer mit welcher Rolle Zugriff hat. Die weiteren Anzeigen sind abhängig von der Rolle im gewählten Modul. Die Rolle ist jeweils übersichtlich im Modul als Tag dargestellt.

## Zusätzliche Features

### 5.2.1 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.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 5: Filter nach Missing Grades und / oder Semester

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 6 : 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. Pro Kurs wird die Durchschnitts-Note der Studenten in der Zeile Average berechnet und blau hinterlegt angezeigt. Zur einfachen Lesbarkeit sind die Namen der jeweiligen Kurse zuunterst tabellarisch noch einmal aufgeführt.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Zur Eingabeunterstützung wird die aktuell selektierte Zeile und Spalte grau hervorgehoben. Somit ist klar, welche Zelle aktuell bearbeitet wird.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

### 5.2.2 Professor

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

### 5.2.3 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 vorausgefüllten Text.

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

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

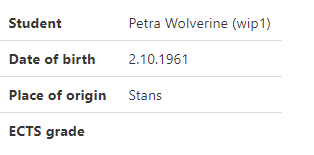
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### 5.2.4 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. Hier haben wir den aktuellen Leistungsausweis der BFH als Inspiration verwendet. Die ECTS-Grade wird wie folgt beim Studenten eingestuft:

* >=90 = A
* >=80 = B
* >=70 = C
* >=60 = D
* >=50 = E
* <50 = F



## Notenberechnung

Der Zwischenerfolg rechnet den Zwischenstand der eingegebenen Noten der Kurse und berücksichtigt die fehlenden Noten nicht. Hingegen rechnet sich der Gesamterfolg aus dem Durchschnitt aller Noten der Kurse. Die Noten werden gerundet.

Die Notenberechnung wird aktuell im Back- und Frontend durchgeführt. Die Berechnung der Noten geschieht wie folgt:

C = Course

S = Student

Der Zwischenerfolg wird analog der obenstehenden Formel berechnet, wobei nur solche Kurse und Noten berücksichtigt werden, wo Noten über 0 eingetragen sind. Sofern alle Noten eingetragen sind, ergeben Gesamt- und Zwischenerfolg das gleiche Resultat.

# 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 war umständlich aber wir haben dadurch unser Wissen erweitert. Der grosse Vorteil von Swagger sind die interaktiven Abfragemöglichkeiten, wodurch sich der Einsatz von Postman eigentlich erübrigt. 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 die REST API auf die SPA optimiert. Dadurch bringt das Design der REST Schnittstelle einige Nachteile mit sich.

Die Projektgruppe befindet, dass alle Ziele erreicht wurden. Für die Projektgruppe war die Aufgabenstellung sehr interessant und ermöglichte es, alle Technologien innerhalb der gelernten Module anzuwenden. Innerhalb der Projektgruppe konnten wir uns auf die unterschiedlichen Stärken der Mitglieder zurückgreifen und viel untereinander lernen.

Zuerst waren wir mit der Aufgabenstellung überrumpelt und wussten gar nicht wo es anzufangen gilt. Nach mehreren Iterationen verständigten wir uns auf das Vorgehen, führten kritische Diskussionen und trafen gemeinsam Entscheidungen. Häufig verfolgten wir den Ansatz, dass alle 4 Studierenden gleichzeitig am selben Problem arbeiteten. Dies war nicht immer der schnellste Weg, jedoch war das Interesse bei allen gross, in alle Bereiche der Software einzusehen und mitzugestalten. So wurden häufig die nächsten Schritte gemeinsam besprochen und verschiedene Wege diskutiert, um das Ziel zu erreichen. Dadurch konnten Erfahrungen und Wissen sehr gut allen zugänglich gemacht werden, Probleme vorausschauend erkannt werden und ein gemeinsamer Wissensstand garantiert werden. Wir haben diese Form der Zusammenarbeit sehr geschätzt und konnten als Gruppe gemeinsam wachsen und voneinander profitieren.

Sehr beeindruckend war für uns der Paradigma-Wechsel von der Multipage-Applikation zur Single-Page-Applikation. Dieser Ansatz war für die meisten unserer Gruppe neu und daher auch herausfordernd. Der Entscheid, welche Daten, wann und wo in der Applikation zur Verfügung stehen müssen und dies von der eigentlichen Datenlogik im Backend zu trennen, war besonders spannend.

# Glossar

Wir haben folgende Begriffsdefinitionen angewendet:

|  |  |
| --- | --- |
| **Module** | CAS, Certificate of Advanced Studies |
| **Course** | Kurs |
| **Grade** | Note pro CAS / Note pro Modul |
| **Rating** | Quote, Erfolgsquote / Einzelnote pro Kurs |
| **Preliminary grade** | Zwischenerfolg |
| **Overall grade** | Gesamterfolgsquote/note |
| **SPA** | Single Page Application |
| **HEAD** | CAS-Verantwortliche/r |