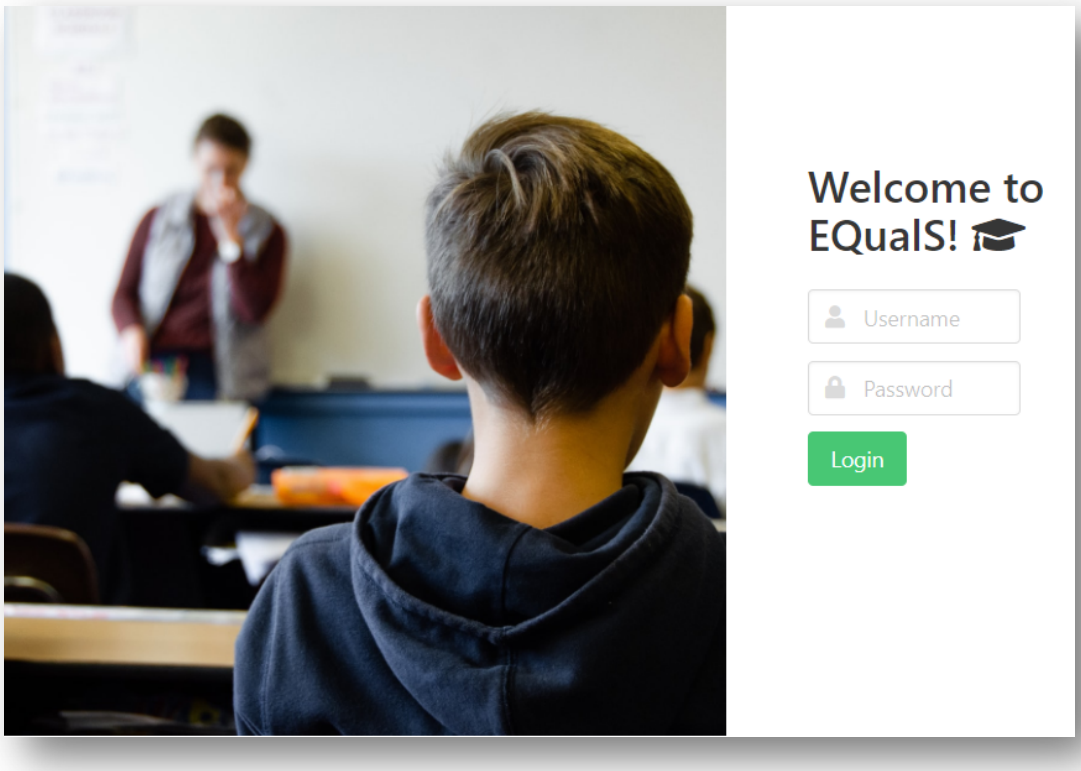
Berner
Fachhochschule

CAS SD: Projekt EQualS
Datum: 25. 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

1.1. Use Case Diagramm

Die Projektgruppe erarbeitete die Projektarbeit auf der Grundlage der Aufgabenstellung:

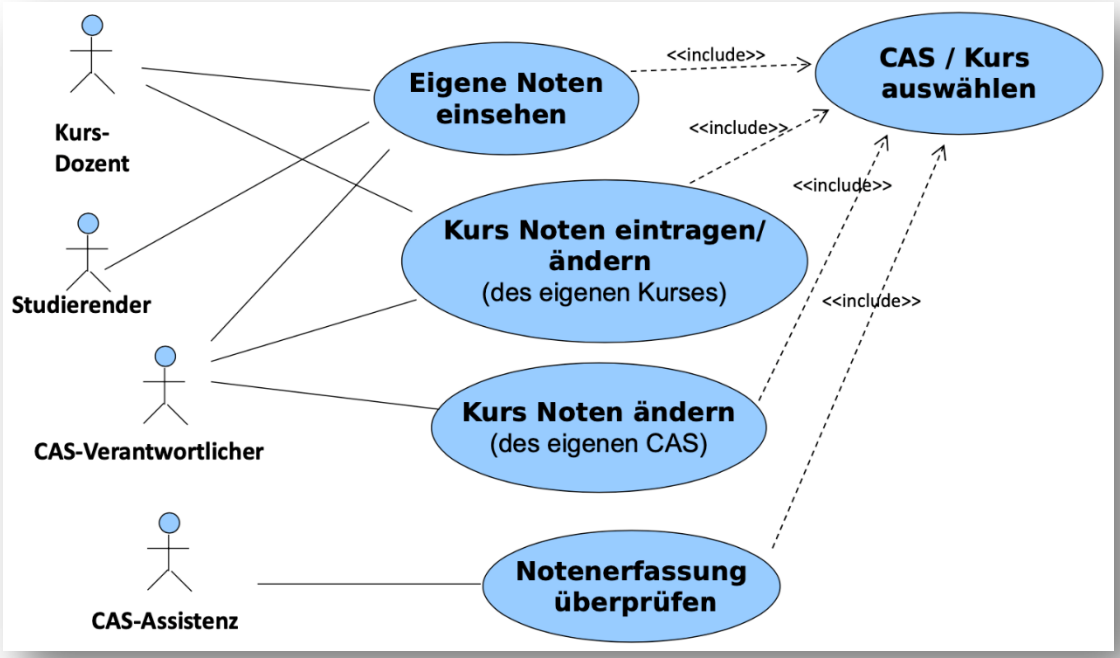


Abbildung 1 Use Case Diagramm

Kurs-Dozenten
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 melden sich in der Applikation an, können dann ein CAS selektieren und die eigenen Kursnoten einsehen.

CAS-Verantwortliche
Nach der Anmeldung in der Applikation, können CAS-Verantwortliche in den CAS, die sie verantworten, Kursnoten aller Studierenden einsehen und ändern. Die CAS-Verantwortlichen können Noten aller Kurse im CAS erfassen. 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 zu machen. Sie haben keine Schreibrechte auf die Noten. Pro CAS gibt es nur eine Assistenz.

Generell:
Es gilt zu beachten, dass eine Person je nach CAS unterschiedliche Rollen haben kann. So ist es zum Beispiel möglich, dass ein Dozent einen Kurs unterrichtet und in einem anderen CAS als Verantwortlicher tätig ist. Auch ist es so möglich, dass eine CAS-Assistenz in einem anderen CAS als Studierende/r eingeschrieben ist.

2. Datenmodell

2.1. 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). Auch wird diese Schwachstelle programmatisch mit Java abgefangen.
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.

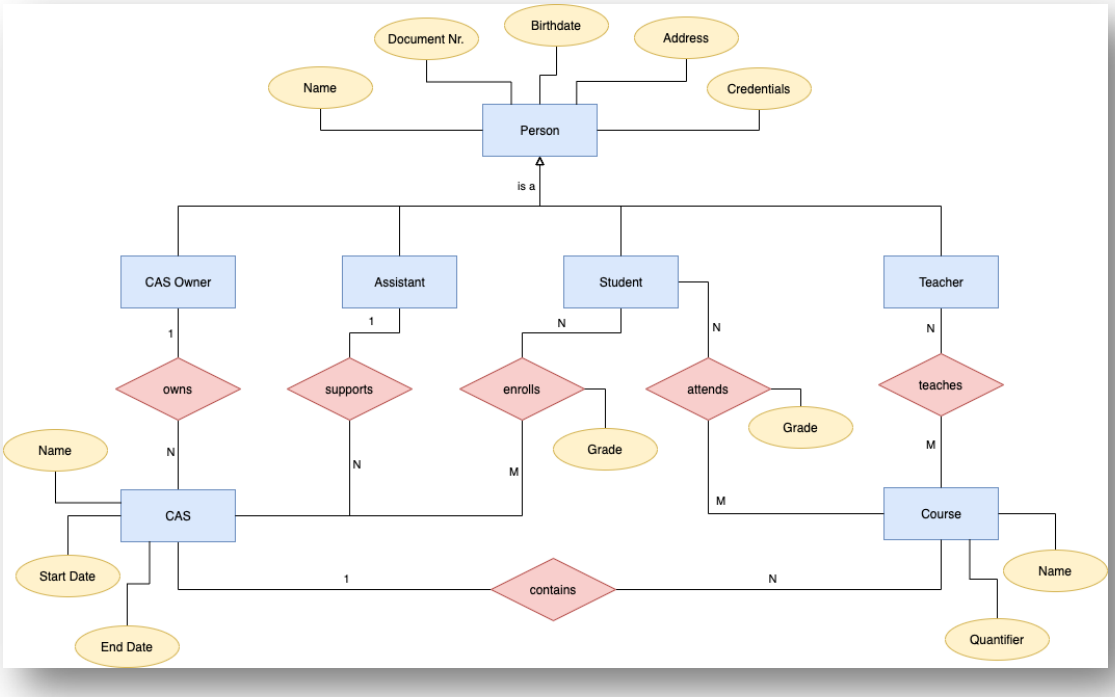


Abbildung 2 ER Datenmodell

2.2. Datenbank-Schema

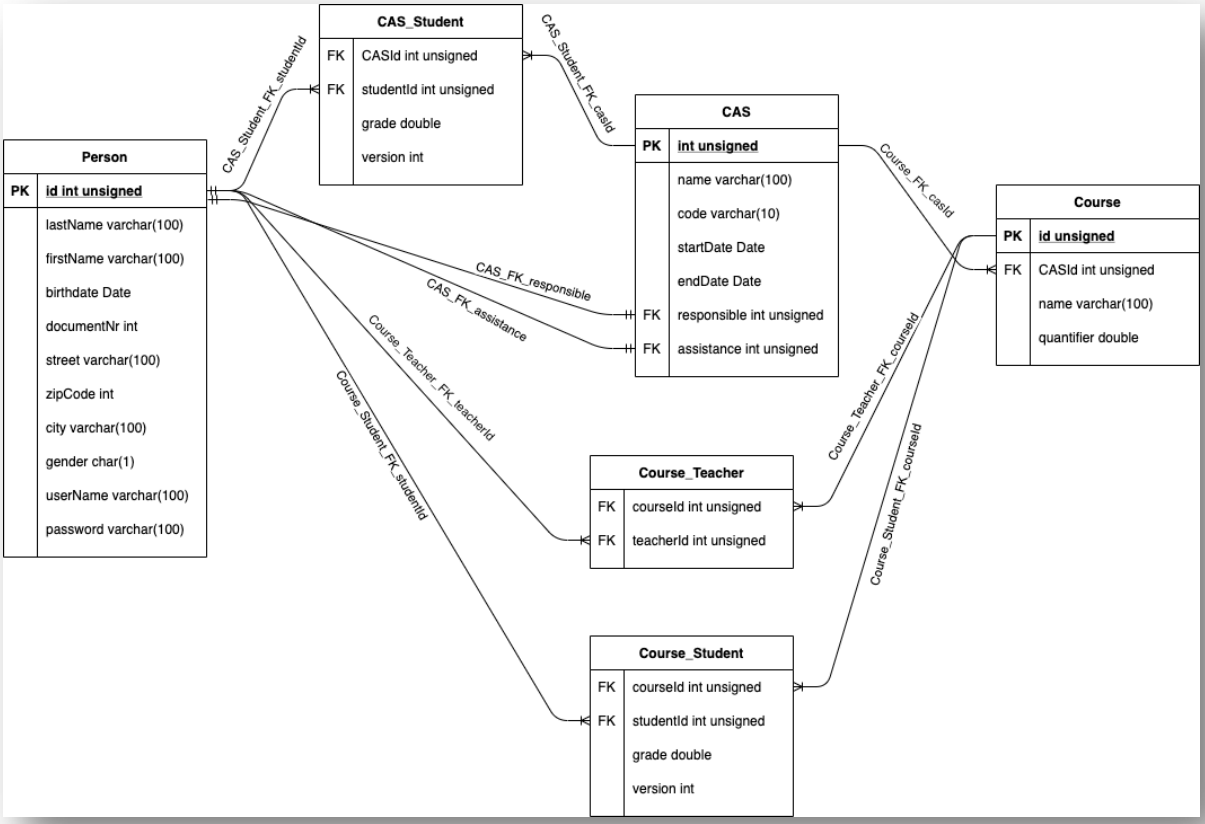


Abbildung 3 DB Schema

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).

2.3. Datenbank-Schema: Konsolidierung

Für den weiteren Verlauf des Projektes wurde den Studierenden ein konsolidiertes DB-Schema abgegeben, auf dessen Basis weitergearbeitet wurde.

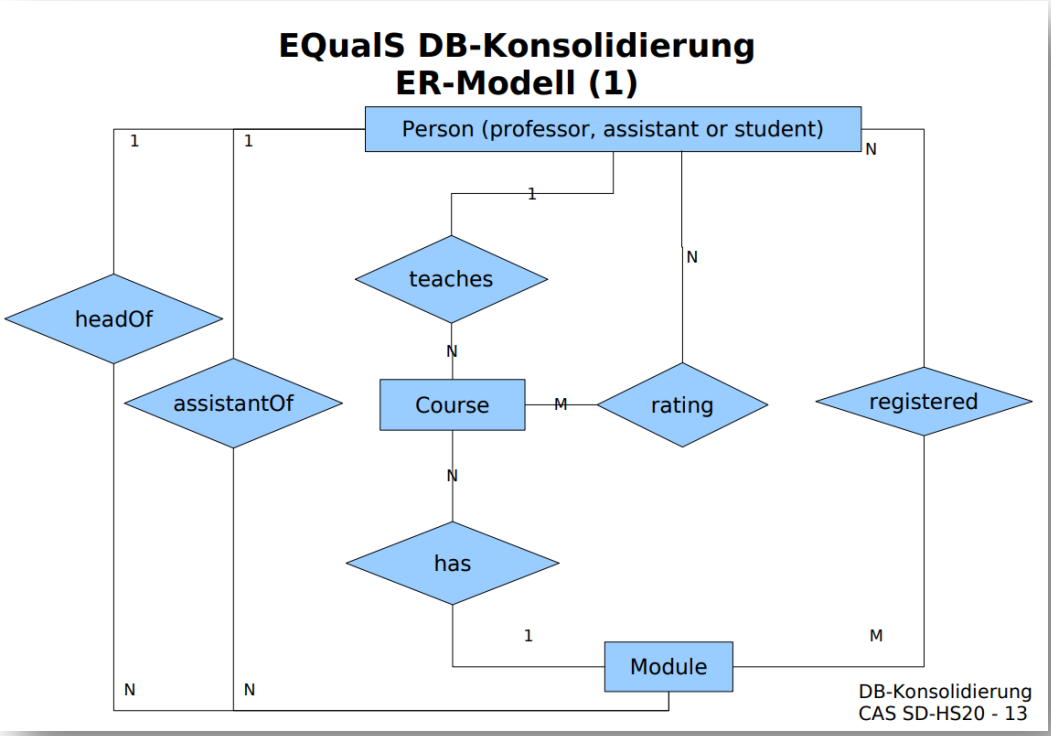


Abbildung 4 DB-Konsolidierung ER-Modell

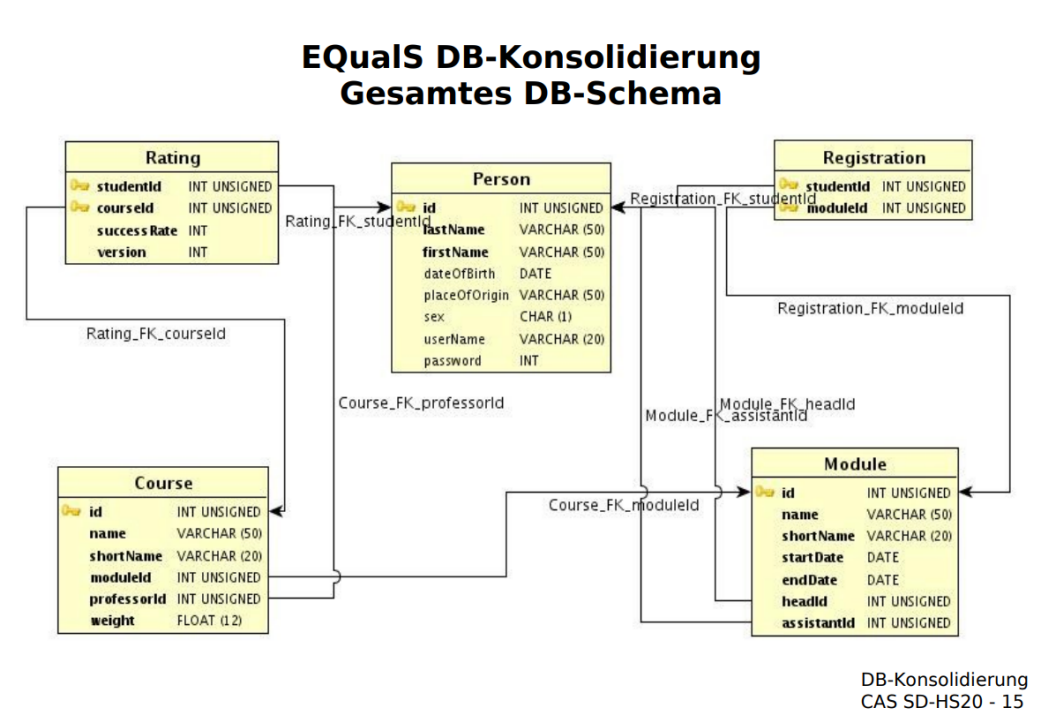


Abbildung 5 DB-Konsolidierung DB-Schema

3. Graphische Benutzeroberfläche

3.1. Ergonomie-Überlegungen

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, auf welche der Benutzer Berechtigungen hat. Module, die zeitlich bereits in der Vergangenheit liegen, werden verblasst dargestellt.

Für die Rollen CAS-Verantwortliche, Dozierende und CAS-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.

3.1.1 CAS-Verantwortliche / Dozierende

Benutzer mit den Rollen CAS-Verantwortlicher oder Dozent können Noten eintragen und verändern. Dem Benutzer CAS-Verantwortlicher werden alle Kurse des Moduls angezeigt. Dem Benutzer Dozent werden nur diejenigen Kurse angezeigt, die er unterrichtet.

3.1.2 CAS-Assistenz

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

3.1.3 Studierende

Studierende können nur ihre Noten von den besuchten CAS einsehen.

3.1.4 Save Button auf der Kursübersicht

Der Save Button wird initial inaktiv angezeigt. Wenn Werte in der Liste verändert werden, wird der Button aktiviert und kann dann betätigt werden. Die Werte werden erst nach dem Betätigen des Save Buttons gesendet und gegebenenfalls gespeichert. Bei der Eingabe der Werte gibt es Formatprüfungen auf die zulässigen Werte. Wenn unzulässige Werte eingegeben werden, kann der Request nicht abgeschickt werden und es wird ein Fehler angezeigt. Nach erfolgter Speicherung erscheint eine Bestätigungsmeldung.

3.2. GUI-Prototyp

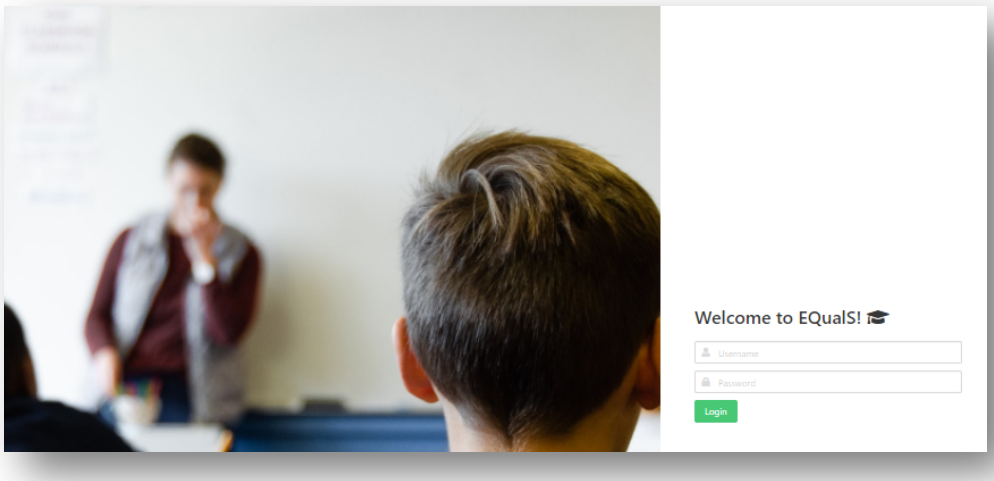


Abbildung 6 Login Mock

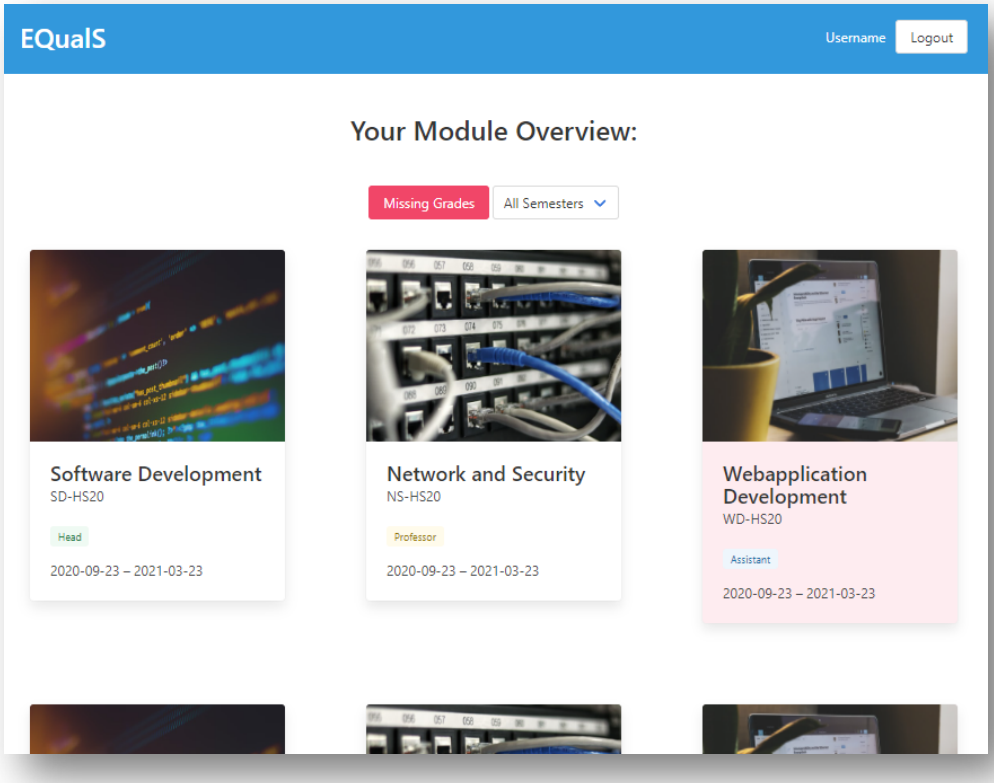


Abbildung 7 Modulübersicht Mock

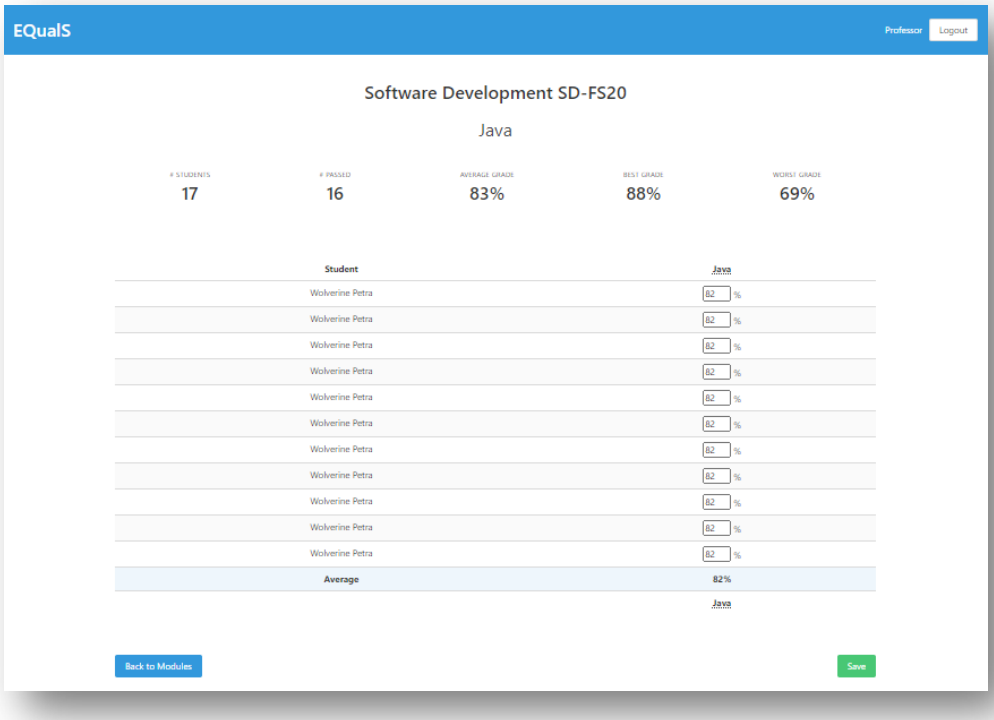


Abbildung 8 Kursübersicht Dozierende Mock

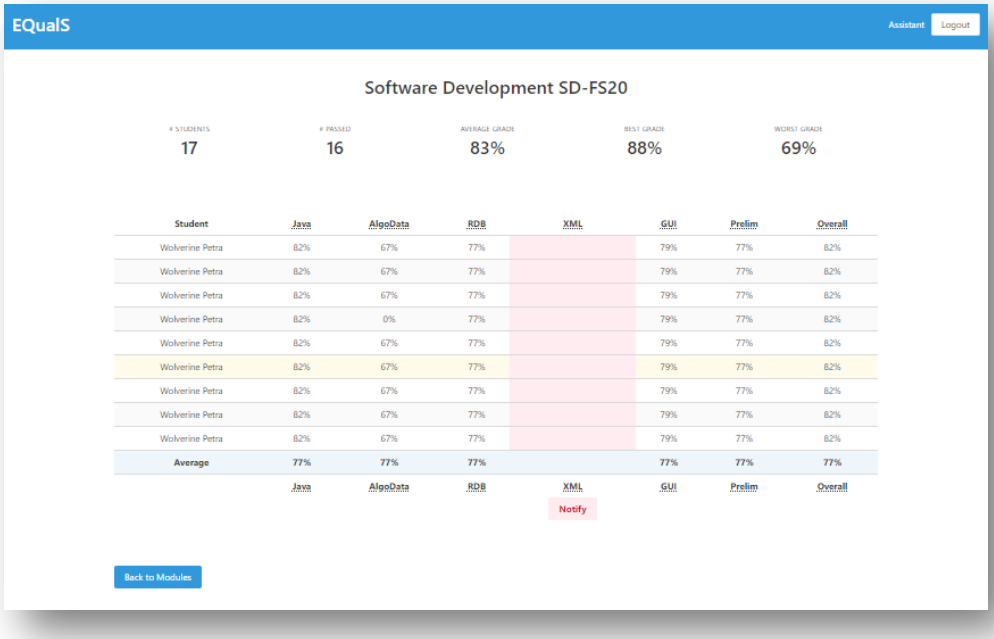


Abbildung 9 Kursübersicht CAS-Assistenz Mock

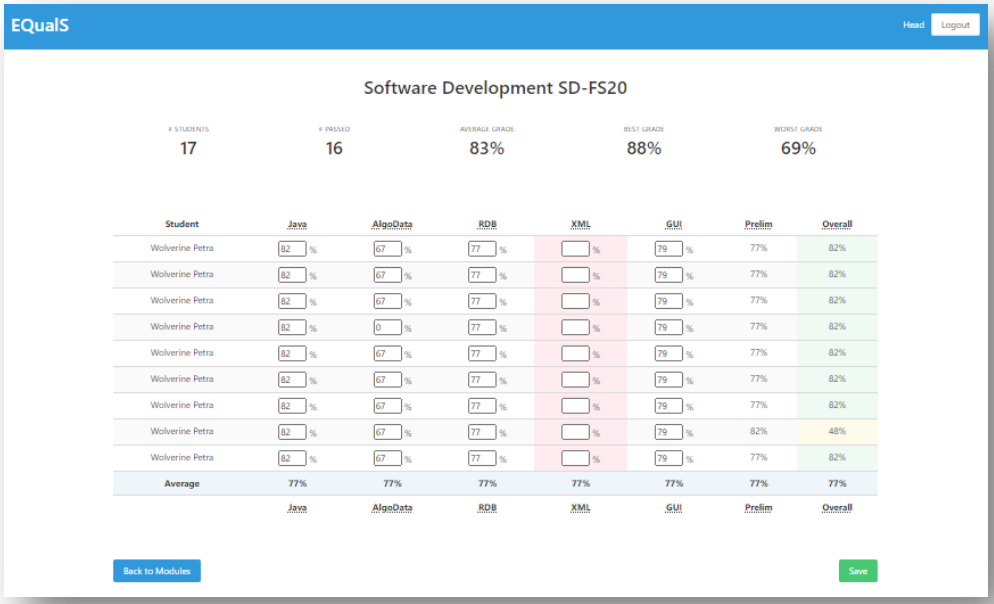


Abbildung 10 Kursübersicht CAS-Verantwortliche Mock

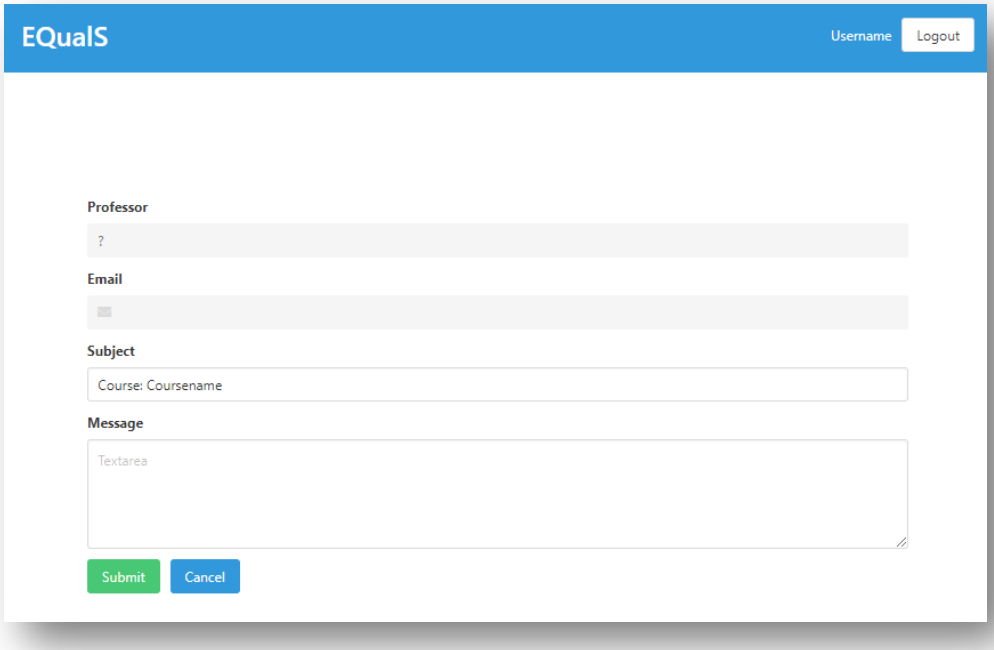


Abbildung 11 Benachrichtigung CAS-Assistenz Mock

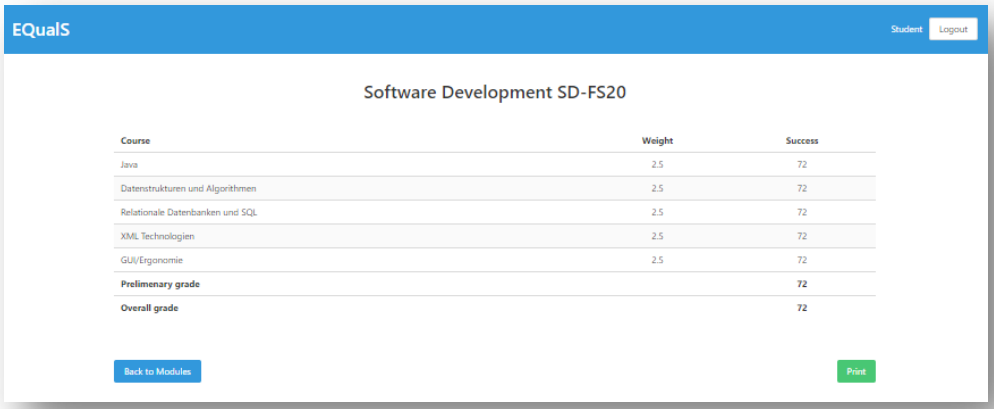


Abbildung 12 Notenübersicht Studierende Mock

3.3. Pageflow

Als zentrale Anlaufstelle in der App dient die Overview-Komponente. Hier können alle Benutzer, unabhängig von der Rolle zwischen den einzelnen Modulen navigieren. Auf einen Blick stehen dem Benutzer alle Module zur Verfügung, wo eine Rolle wahrgenommen wird. Ausser den Studierenden steht allen Benutzerrollen eine Filterung zur Verfügung, weil eine grosse Anzahl Module bei Studierenden nicht erwartet wird. Für CAS-Verantwortliche, Dozierende und CAS-Assistenz wird dieselbe Courses-Komponente aufgerufen, wobei sich die Ansicht je nach Rolle verändert (Anzahl Kurse, Eingabefelder, etc.). Der CAS-Assistenz wird zusätzlich ein Button angeboten, wo bei fehlenden Noten direkt eine Nachricht an den Dozierenden des Kurses gesendet werden kann (E-Mail wird aufgerufen bei Submit). Bei Studierenden wird die Assessment-Komponente aufgerufen, welche eine Notenübersicht des Moduls anzeigt.

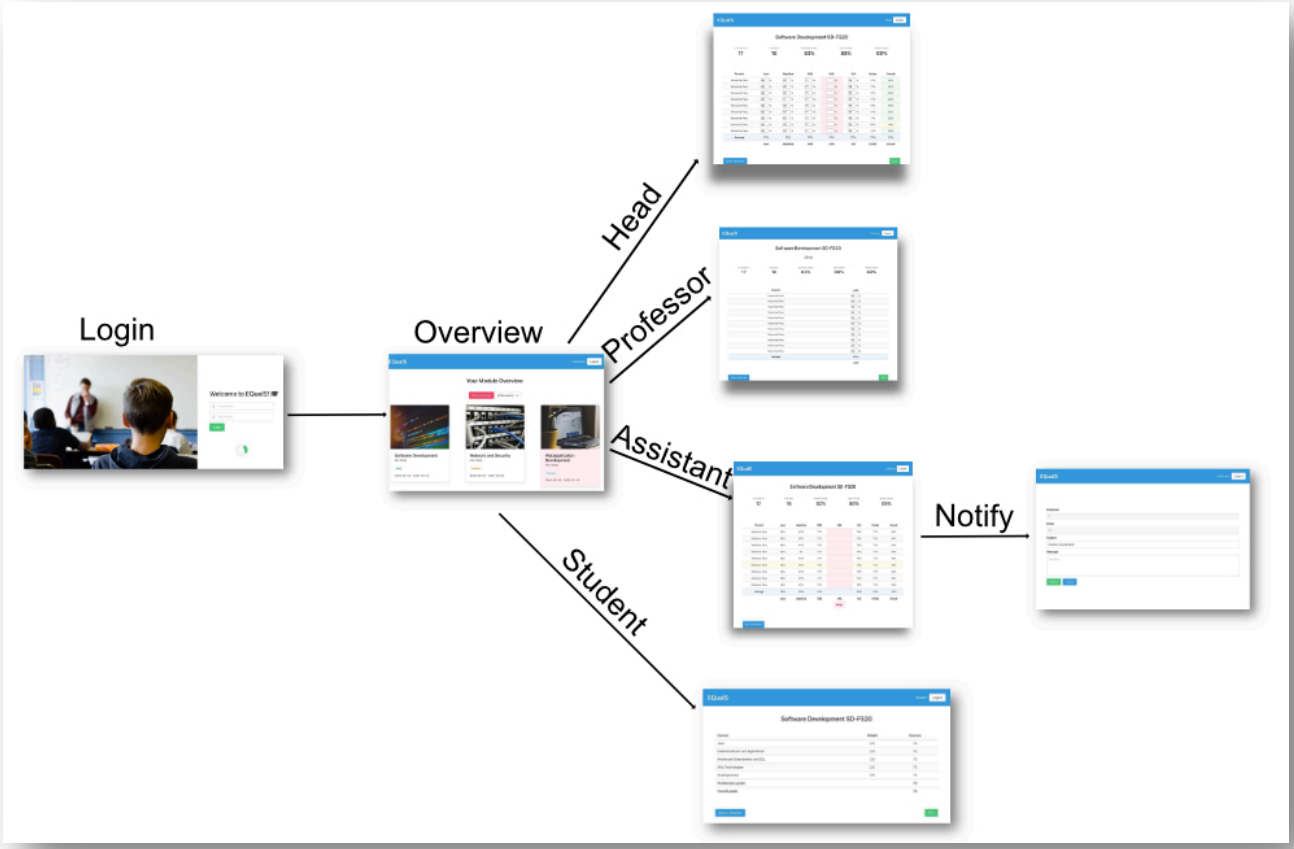


Abbildung 13 Pageflow

4. Gesamtdesign

4.1. 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.

4.2. Architektur

4.2.1 Domänenmodel

Das Domänenmodell ist stark an die Datenstruktur der vorgegebenen 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 Studierenden, Dozierenden, CAS-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 schliesslich 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-Aufrufe.

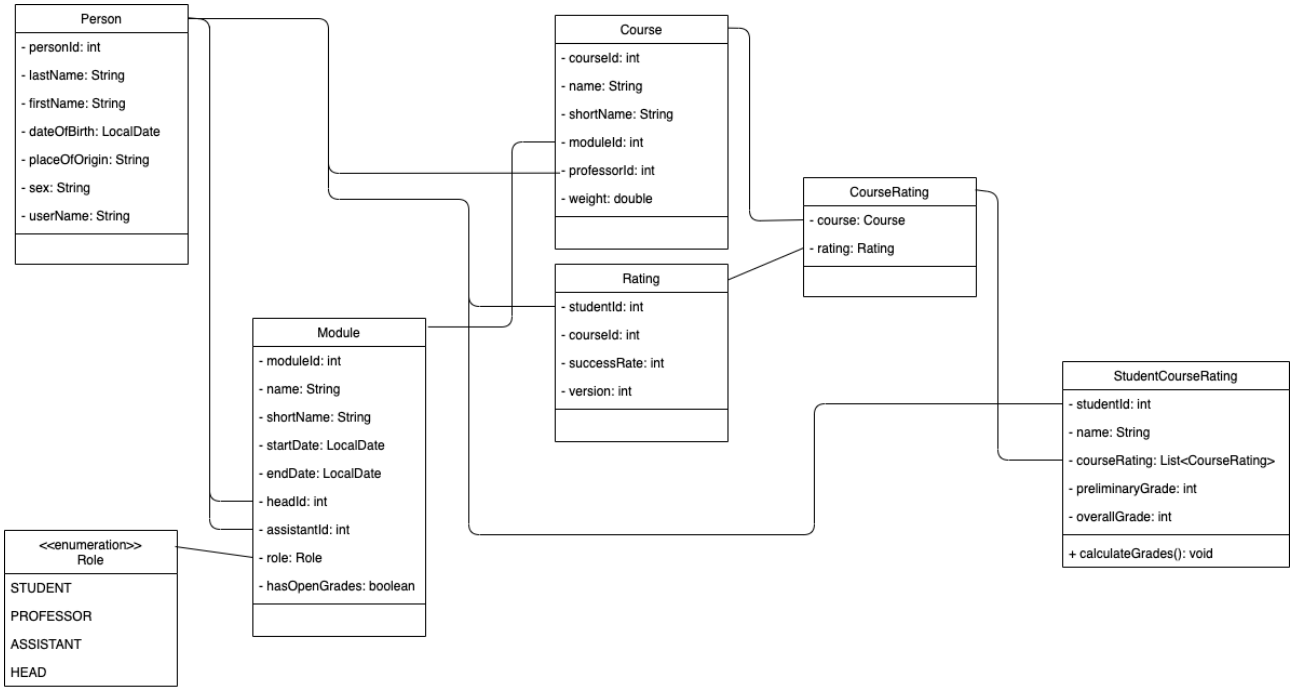


Abbildung 14 Domänenmodell

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 den entsprechenden REST-Controller weiter, wo die Angaben überprüft werden. Die Anfrage wird dann an eine Service-Klasse weitergeleitet, welche die Repository-Schicht aufruft, die 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.

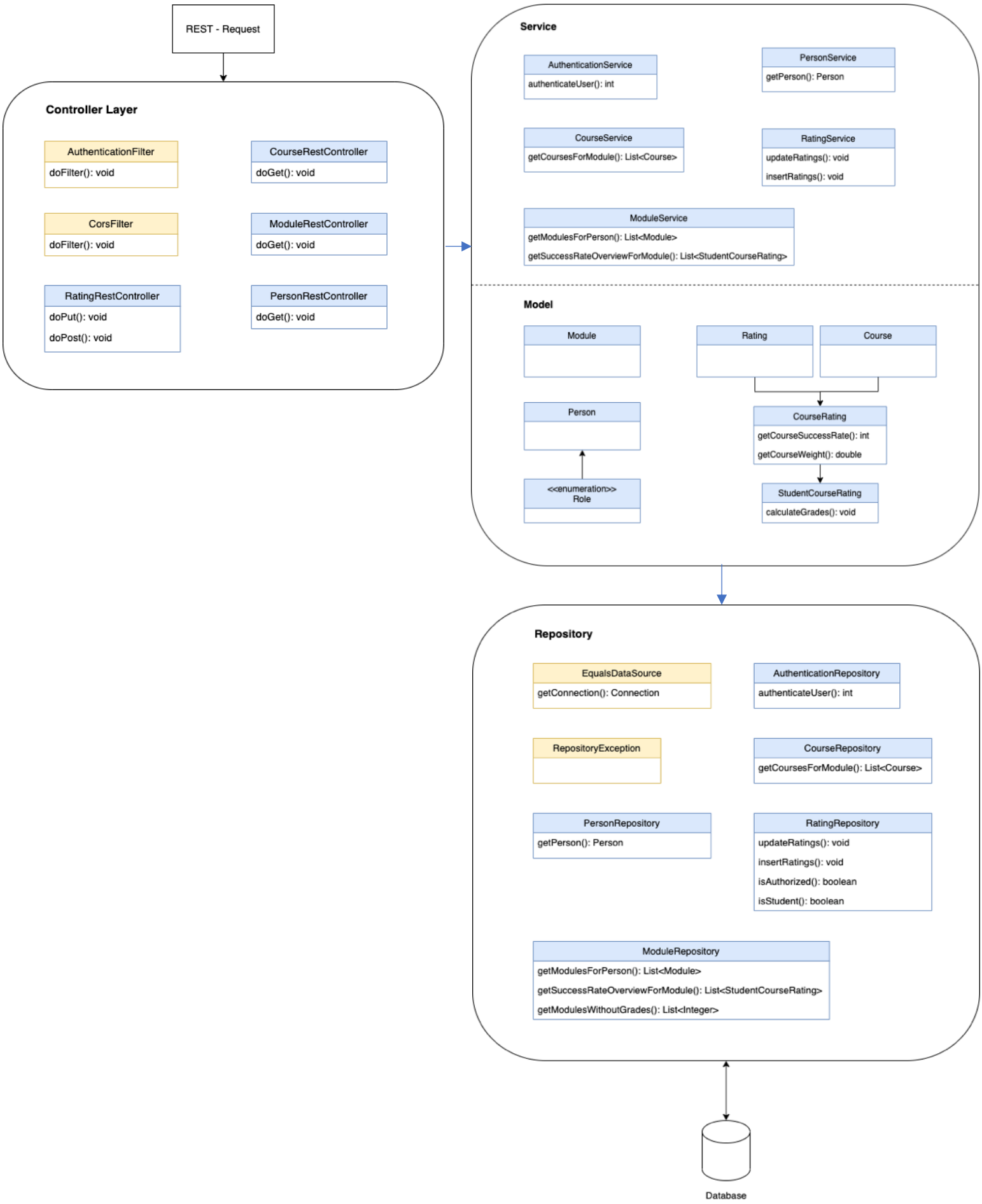


Abbildung 15 Drei-Schichten-Architektur

4.3. REST-Schnittstelle

Das REST API ist über http://localhost:8080/equals_war/api 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*

5. Implementation

Die Designüberlegungen für die Backend-Implementation sind in Kapitel 4 beschrieben. Das Backend wurde in Java implementiert, für das Frontend wurde eine Single Page Applikation in JavaScript erstellt. Für die Umsetzung wurde das CSS-Framework Bulma.io verwendet, welches keine Implementation von JavaScript anbietet. Es steht eine JavaDoc-Dokumentation zur Verfügung, zudem wurden Unit-Tests auf den Businessklassen implementiert. Der Java-Source Code entspricht den vorgegebenen Clean-Code-Richtlinien.
Für den Zugriff auf die Datenbank wurden Prepared-Statements verwendet, was die Gefahr von SQL-Injections minimiert und die Performance verbessert. Um Problemen bei gleichzeitigen Zugriffen auf die Datenbank vorzubeugen, wurde ein Optimistic-Locking implementiert.

5.1. Zugriffskontrolle

Die Zugriffskontrolle erfolgt beim Auslesen der Daten in der Repository-Schicht. Hier wird zunächst geprüft, ob ein Benutzer authentifiziert ist und 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 in der Modulübersicht angezeigt. Zusätzlich hat die Projektgruppe eine Autorisierung eingeführt, welche beim Erfassen bzw. Ändern der Noten angewendet wird. Hier wird jeweils geprüft, ob der Absender des REST-Calls tatsächlich Dozierender oder CAS-Verantwortlicher des mitgeschickten Kurses ist und ob der Studierende im entsprechenden Kurs eingeschrieben ist. Somit sind ungerechtfertigte Manipulationen über REST-Calls nicht möglich.

5.2 Zusätzliche Features

In diesem Kapitel werden zusätzlich eingebaute Features näher erläutert.

5.2.1 Statistiken

Die Kursübersicht zeigt für alle Rollen, ausser für Studierende, Statistiken über das CAS an. Man sieht die Anzahl Studierende, die Anzahl bestandener Studierenden, die Durchschnittsnote sowie die jeweils beste und schlechteste Note. Bei Dozierenden wird die Berechnung auf Basis der gewichteten Kurse durchgeführt.

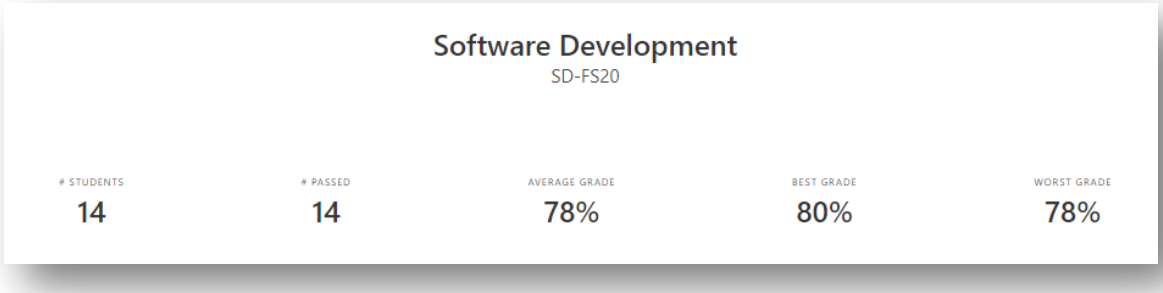


Abbildung 16 Statistiken

5.2.2 CAS-Verantwortliche/r

Wenn für ein CAS noch nicht alle Noten erfasst worden sind, werden diese Module mit der Markierung «Missing grades» ergänzt. Auf der Modulübersichtsseite können die Filter «Missing grades» und Semester angewendet werden. Dies ermöglicht allen Rollen, ausser den Studierenden (aufgrund der zu erwartenden tiefen Anzahl CAS), eine zielgerichtete Filterung. Die Implementierung der Logik erfolgte im Frontend.

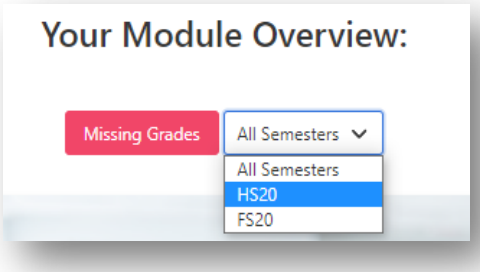


Abbildung 17 Filter nach fehlenden Noten und Semester

Abbildung 18 Darstellung eines CAS in der Modulübersichtsseite inkl. Markierung

Bei der Kursübersichtsseite wird die Overall Spalte grün hinterlegt, wenn das CAS bestanden ist. Ist das CAS nicht bestanden, wird die ganze Zeile mit gelb dargestellt. Somit ist auf einem Blick ersichtlich, welche Studierenden bestanden haben und welche nicht. Pro Kurs wird die Durchschnitts-Note der Studierenden in der Zeile Average berechnet. Zur einfachen Lesbarkeit sind die Namen der jeweiligen Kurse zuunterst tabellarisch noch einmal aufgeführt.

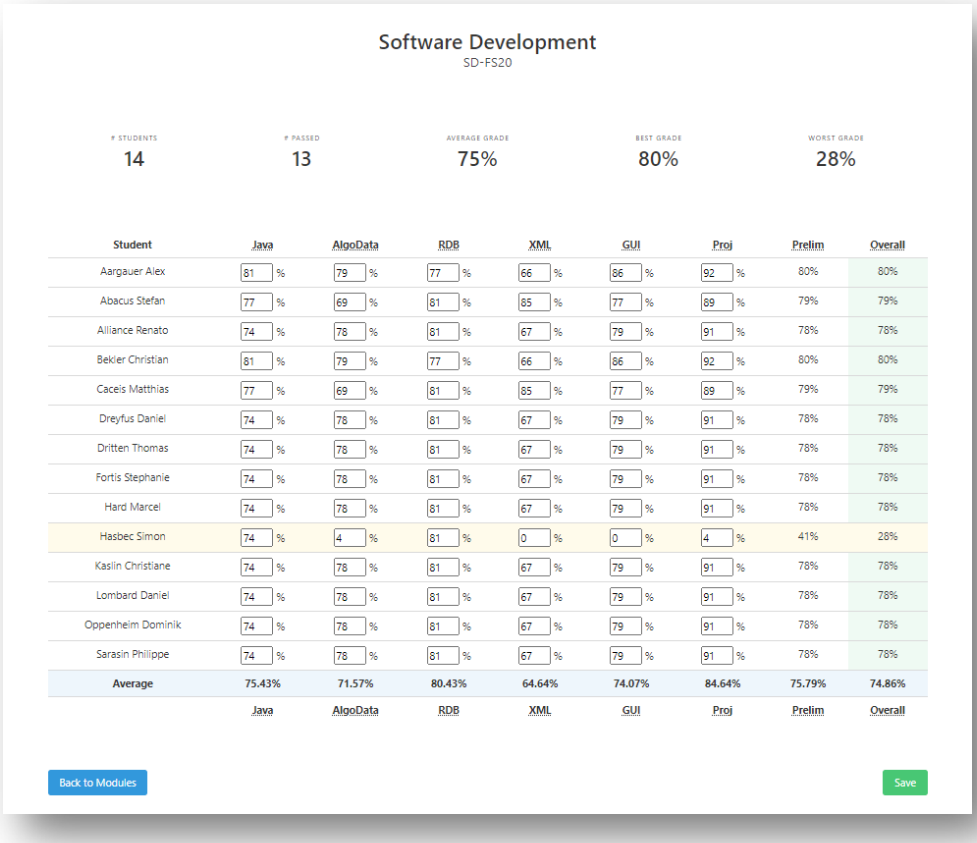


Abbildung 19 Kursübersichtsseite

Zur Eingabeunterstützung wird die aktuell selektierte Zeile und Spalte grau hervorgehoben. Somit ist für den Benutzer eindeutig erkennbar, welche Zelle aktuell bearbeitet wird.

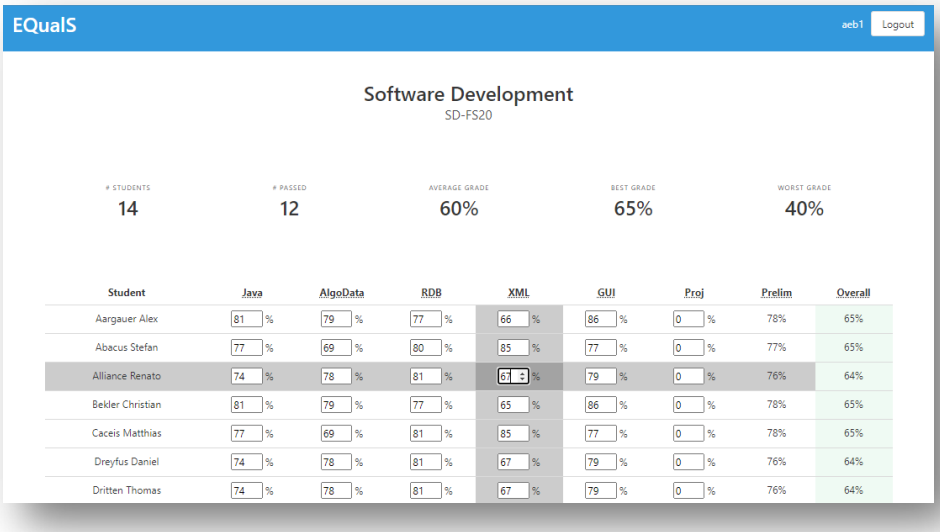


Abbildung 20 Darstellung einer selektierten Zelle

5.2.3 Dozierende

Die Funktionen decken sich mit denen der CAS-Verantwortlichen, ausser, dass nur diejenigen Kurse angezeigt werden, in der der Benutzer auch die Rolle eines Dozierenden einnimmt. Die Kalkulation der Statistiken erfolgt auf Basis der gewichteten Kurse.

5.2.4 CAS-Assistenz

Bei der Kursübersicht werden fehlende Noten rot angezeigt. Die Noten können von der CAS-Assistenz nicht verändert werden. Solange in einem Kurs noch Noten fehlen, kann mittels der Notify-Funktion eine E-Mail mit einem vorausgefüllten Text an den entsprechenden Dozierenden erstellt werden.

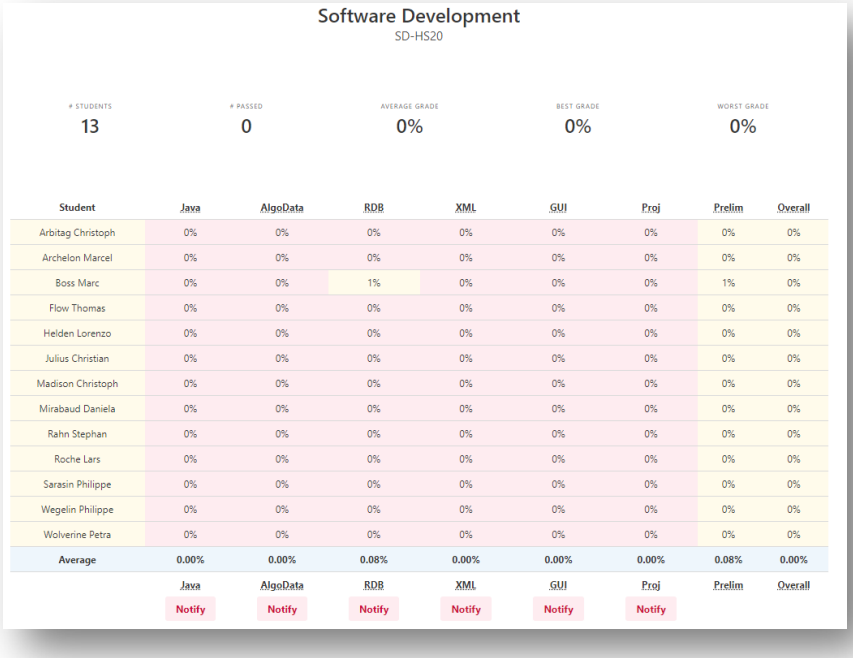


Abbildung 21 Darstellung fehlender Noten inkl. Notify-Funktion

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

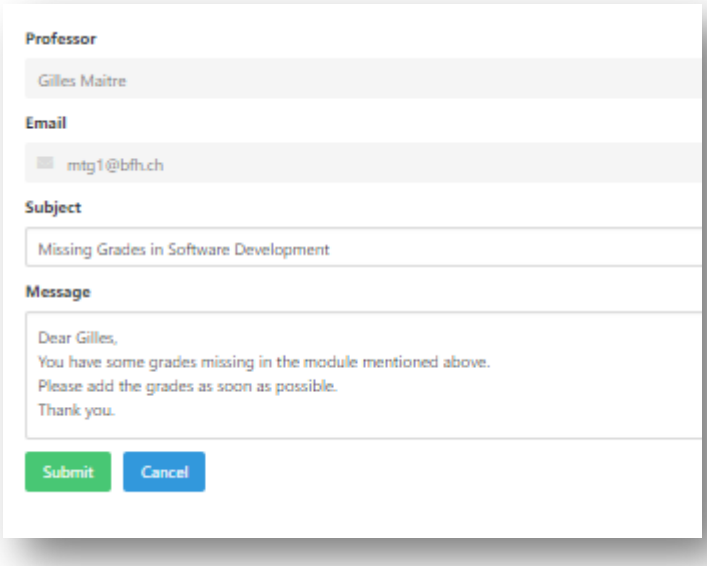


Abbildung 22 Benachrichtigungsmöglichkeit der CAS-Assistenz

5.2.5 Studierende

Studierenden werden die besuchten Module angezeigt. Nach Auswahl eines Moduls sind die darin enthaltenen Kurse, deren Gewichtung, sowie die Noten ersichtlich. Zusätzlich werden weitere Angaben zum Studierenden angezeigt. Hier haben wir den aktuellen Leistungsausweis der BFH als Inspiration verwendet. Die ECTS-Note wird erst berechnet, wenn alle Noten eines CAS eingetragen wurden. Die Einstufung der Studierenden sieht wie folgt aus:

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

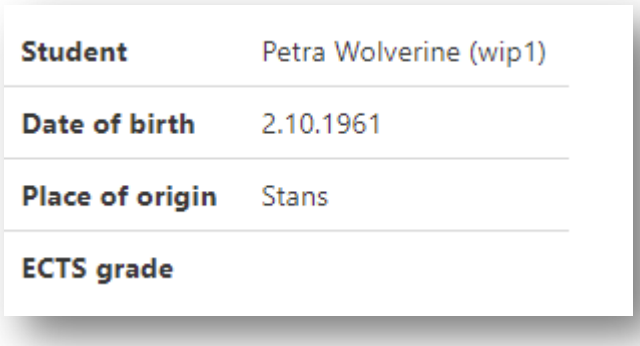


Abbildung 23 Darstellung der Studierenden-Daten im CAS

5.3 Notenberechnung

Der Zwischenerfolg rechnet den Zwischenstand der eingegebenen Kursnoten und berücksichtigt fehlende Noten nicht. Hingegen rechnet sich der Gesamterfolg aus dem gewichteten Durchschnitt aller Kursnoten. Die Noten werden nach den mathematischen Grundregeln gerundet.

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

$$Overall = \frac{\sum(Cweight_1 * Ssuccess_1 + \cdots . + Cweight_n * Ssuccess_n)}{\sum(Cweight_1 + \cdots . + Cweight_n)}$$

$C = Course$
$S = Student$

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

6. Erreichte Ziele und Erfahrungen

Die Projektgruppe ist sich einig, dass alle Ziele erreicht wurden. Für die Projektgruppe war die Aufgabenstellung sehr interessant und ermöglichte es, alle Technologien innerhalb der gelernten Module anzuwenden. Die Projektgruppe konnte auf die unterschiedlichen Stärken der Mitglieder zurückgreifen und viel untereinander lernen. Wir haben während der gesamten Arbeit stets den Ansatz «make it work, make it right, make it fast.» angewendet.

Zuerst waren wir mit der Aufgabenstellung überrumpelt und wussten gar nicht, wo es anzufangen gilt. Nach mehreren Iterationen einigten wir uns auf das Vorgehen, führten kritische Diskussionen und trafen gemeinsam Entscheidungen. Häufig verfolgten wir den Ansatz, dass alle vier Studierenden gleichzeitig mittels Pair-Programming über MS Teams am selben Problem gearbeitet haben. 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 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.
Auch war es für uns lehrreich, Unit-Tests für die Business-Logik zu implementieren. Zwar konnten wir auf Basis unserer Modelle nur wenige sinnvolle Unit-Tests in den Business-Klassen durchführen. Trotzdem war es für uns ein interessanter Prozess, wobei wir auch das Mockito-Framework besser kennengelernt haben.
Die Dokumentation unserer REST API über Swagger war umständlich, aber wir haben dadurch unser Wissen erweitert. Ein grosser Vorteil von Swagger ist die interaktive Abfragemöglichkeit, wodurch sich der Einsatz von Postman eigentlich erübrigt. Erst durch die Arbeit mit Swagger haben wir bemerkt, dass alle Rollen Ratings an unser Backend senden konnten. Dies haben wir danach korrigiert.

Zusammenfassend, sind wir mit der Art der Zusammenarbeit und mit dem erreichten Resultat sehr zufrieden. Wir haben dieses Projekt als einen sehr guten Abschluss für unser CAS erlebt.

7. 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
ECTS	European Credit Transfer System

8. Abbildungsverzeichnis