

Projekt-Schlussbericht

Projekttitel: Gradus

Autor(en): Mirio Eggmann
Nicolas Brechbühler
Manuel Bieri
Dario Menzel

Datum: 13.01.2017

Inhalt

Teil 1	5
Management Summary und Aufgabenstellung	5
Zusammenfassung	5
Ausgangslage.....	5
Ziele	5
Rahmenbedingungen	5
Deklaration der Vorkenntnisse	6
Deklaration der Vorarbeiten und fremden Code	6
Deklaration der verwendeten Firmenstandards.....	6
HERMES 5	6
SCRUM	6
Zeitplan.....	7
Projektplan	7
Zusammenfassung der Projektplanung	8
Arbeitsprotokoll	9
Mirio Eggmann.....	9
Manuel Bieri	11
Nicolas Brechbühler	12
Dario Menzel	13
Teil 2	14
Situationsanalyse.....	14
Ausgangslage.....	14
Stärken.....	14
Schwächen.....	15
Systemziele	16
Lösungsvorschläge.....	17
Variantenübersicht.....	17
Beschreibung der Varianten	17
Bewertung der Varianten (Tabelle)	18
Lösungsbeschreibung	20
Systemarchitektur	22
Systemdesign.....	22
Schnittstellendefinitionen	29
Datenmodel	30
ERD	30
Sicherheit	32
Testkonzept und Testspezifikationen	33
Systemtest - Testspezifikation	33
Einführungskonzept	34

Einführungsplan	34
Migrationsplan	35
Testprotokoll	36
Abnahme	36
Benutzerdokumentation / Anleitung	37
Benutzerhandbuch	37
Integrations- und Installationshandbuch.....	43
Anwender	43
Systeminstallation.....	43
Supporthandbuch	44
Projekterfahrung	45
Teil 3.....	46
Selber erstellte Listings und Skripte	46
Literaturverzeichnis.....	50
Glossar	50
Anhang	51
A. Sourcecode	51

Abbildungsverzeichnis

Abbildung 1 System-Skizze	21
Abbildung 2 Backend Diagramm – Model	22
Abbildung 3 Backend Diagramm - Controller	23
Abbildung 4 Backend Diagramm – Repo	24
Abbildung 5 Frontend Diagramm Teil 1	25
Abbildung 6 Frontend Diagramm Teil 2.....	26
Abbildung 7 Backend Mapping.....	29
Abbildung 8 Frontend Mapping Zugriff	29
Abbildung 9 Hibernate Config	29
Abbildung 10 Hibernate Mapping	30
Abbildung 11 ERD	31
Abbildung 12: Login-Seite	37
Abbildung 13: Registrations-Seite	38
Abbildung 14: Profil-Seite.....	38
Abbildung 15: Seite zum Hinzufügen neuer Noten	39
Abbildung 16: Seite zum Hinzufügen neuer Fächer	40
Abbildung 17: Seite zum Hinzufügen neuer Semester	41
Abbildung 18: Dashboard Übersichts-Seite.....	41
Abbildung 19: Fach Übersichts-Seite	42
Abbildung 20 Backend Filestruktur.....	46
Abbildung 21 Frontend Part 1 Filestruktur	47
Abbildung 22 Frontend Part 2 Filestruktur	48
Abbildung 23 Frontend Part 3 Filestruktur	49

Tabellenverzeichnis

Tabelle 1 Fremder Code	6
Tabelle 2 Projektplan	7
Tabelle 3 Arbeitsprotokoll Mirio Eggmann	9
Tabelle 4 Arbeitsprotokoll Manuel Bieri	11
Tabelle 5 Arbeitsprotokoll Nicolas Brechbühler	12
Tabelle 6 Arbeitsprotokoll Dario Menzel	13
Tabelle 7 Stärken der bestehenden Lösung.....	14
Tabelle 8 Schwächen der bestehenden Lösung.....	15
Tabelle 9 Ziele	16
Tabelle 10 Variantenübersicht.....	17
Tabelle 11 Variante a	18
Tabelle 12 Variante b	18
Tabelle 13 Variante c	19
Tabelle 14 Variante d.....	19
Tabelle 15 Abgedeckte Anforderungen mit der Lösungsbeschreibung.....	21
Tabelle 16 Beschreibung der Elemente - Backend.....	27
Tabelle 17 Beschreibung der Elemente - Frontend	28
Tabelle 18 ERD Beschreibung	30
Tabelle 19 Informationssicherheit	32
Tabelle 20 Datenschutz	32
Tabelle 21 Testfälle.....	33
Tabelle 22 Kontakte Migration.....	35
Tabelle 23 Testprotokoll.....	36
Tabelle 24 Abnahme	36
Tabelle 25 Benötigte Software	43
Tabelle 26 Glossar	50

Teil 1

Management Summary und Aufgabenstellung

Zusammenfassung

Im Rahmen des Modul 306 mussten wir in einer kleinen Gruppe ein Projekt durchführen. Unser Team hat sich entschieden ein Notenerfassungstool namens «Gradus» zu programmieren. Dieser Bericht ist Teil von unserer Projektarbeit und stellt den Schlussbericht dar. Dieser Bericht fasst grundsätzlich das gesamte Projekt und alle zuvor erstellten Dokumente in einem Dokument zusammen. Er beschreibt unter anderem unsere Vorkenntnisse, Vorarbeiten, Organisation und Erfahrungen der Gruppenarbeit. Weiter beinhaltet das Dokument auch alle Angaben zum Produkt. Die Produktbeschreibung umfasst entscheidungstechnische Angaben, Systemdiagramme, Handbücher und weitere Angaben, welche zur Weiterentwicklung, Wartung oder zum Verstehen des Produktes benötigt werden.

Ausgangslage

In der aktuellen Situation ist es sehr umständlich alle seine Noten, mit einem Programm überall komfortabel eintragen zu können. Daher muss man sie meistens mit mehreren Programmen verwalten und das ist eher mühsam.

In der neuen Lösung geht es darum, dass der Benutzer seine Schulnoten komfortabel verwalten kann. Noten können eingetragen, bearbeitet und gelöscht werden. Darüber hinaus hat der Benutzer all seine Noten immer schön im Überblick. Die Problemstellung dabei ist es, die Applikation den Wünschen und Bedürfnissen der Benutzerpalette so anzupassen, dass der Grossteil der User zufrieden ist. Sei es das User Interface oder der Ablauf innerhalb des Programms. Die Applikation soll stets verfügbar sein, auch unterwegs auf dem Smartphone. Als Vorarbeit haben sich einige Teammitglieder bereits in die Technologien eingearbeitet, welche neu für sie waren.

Ziele

- Das Tool ist auf allen gängigen Plattformen verfügbar und ermöglicht somit die einfache Verwaltung von Noten.
- Das Tool ist auf das schweizerische Schulsystem ausgelegt.
- Das Tool gibt einen guten Überblick über die eigenen Noten und unterstützt die Berufsbildner bei der Kontrolle der Noten seiner Lehrlinge.

Rahmenbedingungen

- Jeden Dienstagmorgen, bevor mit den Teilaufträgen am Projekt weitergefahren wird, wird in fünf bis zehn Minuten der erzielte Fortschritt und wie es weitergeht für die nächsten Tage besprochen. Ebenfalls wird auch angeschaut, ob jeder noch genug zu tun hat.
- Jeden Dienstag von 08:00 bis 11:30 Uhr arbeiten alle Teammitglieder gemeinsam am Projekt weiter.
- Am Schluss der drei Arbeitsstunden, welche jeden Dienstagmorgen zur Verfügung stehen, wird besprochen, welche Hausaufgaben jedes Teammitglied erledigt, wenn es welche fertigzustellen gibt. Auf diese Weise wird sich das Team in der Woche darauf nicht mehr mit alten, langwierigen Dingen beschäftigen, sondern könnte gleich mit den nächsten Aufgaben weiterfahren.

Deklaration der Vorkenntnisse

Im Projekt Gradus waren unter den Teammitgliedern sehr unterschiedliche Vorkenntnisse vorhanden. Für die Applikation wurden diverse Technologien verwendet. Unter anderem Java Spring, Angular 2, MySQL. Zuvor wurde noch Angular 1 eingesetzt, jedoch haben wir noch von Angular 1 auf Angular 2 migriert. Manuel Bieri hatte vom Betrieb aus schon gute Kenntnisse im Bereich Java Spring Backend, in MySQL und in Angular 1. Angular 2 war für Manuel Bieri noch neu. Mirio Eggmann hat bereits mit Java gearbeitet und MySQL als Datenbank verwendet. Jedoch noch keine Kenntnisse in Java Spring, Angular 1 & Angular 2 erworben. Dario Menzel und Nicolas Brechbühler hatten ebenfalls bereits mit Java und MySQL gearbeitet, jedoch nur in der Schule. Von den anderen Bereichen hatten sie noch gar keine Erfahrung. HERMES wurde mit ein paar Ansätzen von Scrum als Projektmanagementmethode verwendet. Alle im Team hatten bereits einmal mit HERMES in der Schule gearbeitet, jedoch noch nie wirklich produktiv eingesetzt, ausser Dario Menzel, der HERMES in seinem Betrieb täglich benutzt. Somit hatten wir in diesem Bereich einen Spezialisten, der wusste was bei den einzelnen Punkten verlangt wird.

Deklaration der Vorarbeiten und fremden Code

Für Gradus wurden keine grossen Vorarbeiten gemacht. Es wurde eine WhatsApp Gruppe erstellt, um innerhalb vom Team einfach und unkompliziert kommunizieren zu können. Weiter hat sich Mirio Eggmann im Internet ein paar Online Tutorials zu Angular 2 angeschaut. Mit den Ausbildnern wurde innerhalb der Post noch abgesprochen ob eine allfällige Migration auf unser System möglich wäre. Herausgekommen ist, dass es grundsätzlich schon möglich wäre, aber sehr aufwändig wäre und vermutlich zu wenig Nutzen bringen würde für den Aufwand.

Fremder Code

Tabelle 1 Fremder Code

	Beschreibung
Angular2-mdl	Ein Angular2 Modul, welches es erlaubt vordefinierte Material Komponenten zu verwenden. Wurde im Frontend Bereich eingesetzt.
Angular2	Ein JavaScript basiertes Frontend Framework. Wurde in Gradus für das gesamte Frontend verwendet.
Spring	Ein Java Framework. Wurde in Gradus für das gesamte Backend verwendet.

Deklaration der verwendeten Firmenstandards

Als Projektmanagementmethode haben wir HERMES 5 mit einem SCRUM Ansatz verwendet, wie vom Modul 306 vorgeschrieben.

HERMES 5

HERMES 5 wird von der schweizerischen Bundesverwaltung entwickelt und ist ein offener Projektmanagement Standard der in sehr vielen Projekten verwendet wird. Es ist einfach und verständlich. Es gibt eine klare Aufgabenbeschreibung, konkrete Rollenbeschreibungen und Dokumentenvorlagen für schnelle Ergebnisse. Es unterstützt Fachspezialisten bei der Projektausführung, den Projektleiter bei der Planung, Kontrolle und Führung, das Management in der Steuerung.

SCRUM

SCRUM ist eine agile Methode zur Projektentwicklung. Im Modul 306 haben wir grundsätzlich mit HERMES 5 gearbeitet, jedoch auch einige Ansätze von SCRUM angewendet. Zum Beispiel wurden Daily SCRUMs durchgeführt, jedoch war keine Person spezifisch ein SCRUM Master. Weiter wurde das Product Backlog und die Sprints von SCRUM angewendet.

Zusammenfassung der Projektplanung

Grundsätzlich haben wir unser Projekt relativ gut geplant. Jedoch hat es immer wieder kleine Störungen gegeben z.B.:

- Jemand wurde krank.
- Wir mussten noch ein Teamjournal schreiben und sind daher nicht zur Projektarbeit gekommen
- Wir haben die Technologie gewechselt und dementsprechend mehr Aufwand gehabt

Initialisierungsphase

Bei der Initialisierungsphase hatten wir keine grosse Mühe. Wir wussten jedoch noch nicht, wie genau das Dokument bewertet wird und waren daher unsicher ob es gut ist, wie wir es gemacht haben. Das Resultat ist jedoch gut rausgekommen.

Konzeptphase

Die Konzeptphase ist relativ reibungslos verlaufen. Wir konnten den Bericht gemeinsam, vollständig zum Abschluss bringen ohne viel zusätzlichen Aufwand zuhause.

Realisierungsphase

In dieser Phase hatten wir das Problem, dass die Entwickler in unserem Team ein paar Mal abwesend waren und daher kam das Produkt sehr schleppend voran. Daher kamen wir in dieser Phase sehr in Verzug, weil die Dokumentation auch nicht weitergemacht werden konnte, weil die nötigen Elemente vom Programm gefehlt haben, um ein Klassendiagramm, ERD oder ähnliches zu erstellen. Durch den Zeitmangel mussten wir im grafischen etwas improvisieren, jedoch ist es dennoch akzeptabel. Schlussendlich mussten wir sehr viel noch zuhause erledigen, um die Realisierung zu einem Ende zu bringen.

Einführungsphase

Diese Phase dauerte nur einen und daher war der Aufwand des Dokumentes auch kleiner. Jedoch war Mirio Eggmann der einzige der daran arbeiten konnte, weil die anderen Teammitglieder noch das Teamjournal 4 fertigstellen mussten.

Abschlussphase

Diese Phase schliessen wir nun mit diesem Dokument ab. Es beinhaltet dieses Dokument (der Schlussbericht) und die Präsentation / Demonstration vom Produkt. Für diese Phase stand uns nur ein Tag in der Schule zur Verfügung und daher eher unrealistisch das man in dieser Zeit fertig wird. Daher mussten wir nun in den Ferien den Bericht fertigstellen und auch die Präsentation so weit wie möglich vorbereiten.

Arbeitsprotokoll

Mirio Eggmann

Tabelle 3 Arbeitsprotokoll Mirio Eggmann

Arbeitsjournal Projekt: gradus Klasse: inf2014.5g Mirio Eggmann Datum: 29.11.2016			
Tatsächlicher Zeitbedarf	Geplanter Zeitbedarf	Beschreibung der Arbeit	Bemerkungen, Probleme, genutzte Hilfestellungen
0.5h	0.5h	Webserver aufsetzen um am Ende die Applikation Online zur Verfügung zu stellen	<p>Einen Webserver für unser Projekt Gradus aufgesetzt. Dies auf der folgenden Seite:</p> <p>https://www.digitalocean.com/</p> <p>Einen Webserver mit 512MB Ram, 20GB Speicher und Ubuntu 16.04 als OS.</p>
1.5h	1h	Verbindung Angular 2 Frontend mit dem Java Spring Backend	<p>Folgende Ressource haben Manuel und ich als Inspiration für die Verbindung zwischen Backend und Frontend verwendet:</p> <p>https://github.com/borysn/spring-boot-angular2</p>
0.1h	0.2h	ERD aktualisieren	
1.5h	1h	Frontend Angular 2 Views für Login, Registrieren, Profile aktualisieren.	<p>Für das Grid System in MDL habe ich folgende Ressource benötigt:</p> <p>https://github.com/google/material-design-lite</p> <p>https://webdesign.tutsplus.com/tutorials/learning-material-design-lite-the-grid--cms-24531</p> <p>Bemerkung: In weiteren Projekten nie wieder MDL verwenden, das Framework ist sehr unübersichtlich und nicht gut zu gebrauchen.</p>

0.2h	0.2h	Github Repository aufräumen	Einen neuen Branch machen, damit Manuel und ich uns nicht in die Quere kommen. Gerade bei Angular 2 muss man gut aufpassen, dass man nicht gleichzeitig in den gleichen Files arbeitet, weil sonst die Fehlersuche mühsam werden kann.
------	------	-----------------------------	--

Reflexion des Arbeitstages

Der Arbeitstag heute ist relativ gut verlaufen. Ich bin gut vorwärtsgekommen, da ich aber mit Manuel Schwierigkeiten hatte das Angular 2 Frontend mit dem Spring Backend zu verbinden habe ich etwas Zeit verloren. Durch den Wechsel von Angular 1 auf Angular 2 haben wir tatsächlich etwas Zeit verloren. Jedoch finde ich, dass es sich gelohnt hat, denn nun lernen wir noch ein neues Framework, welches in Zukunft und schon heute viel verwendet wird und somit ist unsere Applikation auf dem neusten Stand. Das ERD habe ich noch mit den besprochenen Änderungen aktualisiert. Im Github Repository musste ich prüfen, dass keine Inkonsistenzen entstehen. Dies weil ich und Manuel nun eine kurze Zeit beide im Frontend arbeiten und somit kann es auch passieren das wir in den selben Files arbeiten. Daher erstellte ich einen weiteren Branch, damit wir am Ende beim Mergen prüfen können, ob unser Code kompatibel ist. Einen Webserver habe ich auch bezogen auf DigitalOcean und diesen konfiguriert. Dies hat mit der Zeitplanung gut überein- gestimmt, denn ich habe gerade eine halbe Stunde gebraucht. Weiter habe ich dann noch das Angular 2 Frontend einiger Views aktualisiert, da wir diese erst im Angular 1 hatten und diese nun komplett anders sind. Da wir MDL als CSS Framework benutzen, ist das Designen der Views etwas mühsam geworden, jedoch hat es am Schluss trotzdem geklappt.

Pendenzen für den nächsten Arbeitstag

Zuhause werde ich nun das Frontend von unserer Applikation fertigstellen. Insbesondere unser Dashboard, welches das Herzstück der Anwendung ist. Das Frontend werde ich dann wiederum mit dem Backend binden, dafür wird mir Manuel vom Backend die entsprechenden Daten zur Verfügung stellen. Weiter werde ich mit Manuel ein Installationsscript erstellen, welches das automatische installieren & ausführen der Applikation auf dem Webserver ermöglicht. Im Realisierungsbericht werde ich alle Konfigurationen für das Frontend nachtragen und entsprechende Diagramme für meinen Teil erstellen und einfügen.

Manuel Bieri

Tabelle 4 Arbeitsprotokoll Manuel Bieri

Arbeitsjournal Projekt: gradus Klasse: inf2014.5g Manuel Bieri Datum: 06.12.2016			
Tatsächlicher Zeitbedarf	Geplanter Zeitbedarf	Beschreibung der Arbeit	Bemerkungen, Probleme, genutzte Hilfestellungen
30 min	30 min	Backend Cleanup	
1.5h	1h	Model First – Datenstruktur verbessert	Viel Codeleichen, verschiedene Änderungen waren nötig
1h	1h	Frontend – Design Änderungen	Design Framework nicht optimal für meine Wünsche
Reflexion des Arbeitstages <p>Heute war ich langsamer als geplant. Aufgrund der Model First Änderungen (Verschiebungen, Überschreibung und Zusammenfügungen) mussten andere Codestellen geändert werden. Dies erzeugte eine kurzzeitige Unmöglichkeit, einen Build zu erstellen...</p> <p>Mit einem Aufwand von einer halben Stunde konnte dieses Problem behoben werden.</p>			
Pendenzen für den nächsten Arbeitstag <p>Durch die Änderungen am Model, wurde die DB Struktur verändert. Nun muss die DB aktualisiert werden.</p> <p>Danach setze ich mich an die Bereinigung von Angular 2 dies sollte aufgeräumt werden.</p>			

Nicolas Brechbühler

Tabelle 5 Arbeitsprotokoll Nicolas Brechbühler

Arbeitsjournal Projekt: gradus Klasse: inf2014.5g Nicolas Brechbühler Datum: 06.12.2016			
Tatsächlicher Zeitbedarf	Geplanter Zeitbedarf	Beschreibung der Arbeit	Bemerkungen, Probleme, genutzte Hilfestellungen
5min	10min	Sprintreview	Storyboard
20min	10min	Einrichtung Dropbox und Slack	Ich habe meinen bestehenden Dropbox-Account reaktiviert und bin der Arbeitsgruppe beigetreten. Für Slack habe ich einen neuen Account erstellt.
30min	30min	Arbeitsjournal führen	Als Vorlage konnte dieses Dokument vom Klassenshare verwendet werden.
2h	2h	Realisierungsbericht schreiben	Als Vorlage konnte ein Dokument vom Klassenshare verwendet werden.
5min	10min	Abgabe Dokumente	Dokumente an Lehrperson mailen
Reflexion des Arbeitstages <p>Mit den heutigen Arbeitszielen bin ich gut vorwärtsgekommen. Der geplante Zeitbedarf hat ziemlich gut übereingestimmt mit dem gesamten Zeitbedarf. Jedoch bin ich mit dem Realisierungsbericht noch nicht ganz fertig, was etwas schade ist. Glücklicherweise haben wir dafür aber noch etwas Zeit.</p>			
Pendenzen für den nächsten Arbeitstag <p>Da wir in unserer Gruppe mit der Arbeit «Realisierungsbericht schreiben» noch nicht ganz fertig sind, muss dies auf jeden Fall bis spätestens nächsten Sonntag noch erledigt werden. Ausserdem muss das Teamjournal auf nächste Woche noch erstellt werden. Was ich dann am nächsten Arbeitstag tatsächlich erledigen werde, zeigt sich nach dem Sprintreview nächster Woche.</p>			

Dario Menzel

Tabelle 6 Arbeitsprotokoll Dario Menzel

Arbeitsjournal Projekt: gradus Klasse: inf2014.5g Dario Menzel Datum: 06.12.2016			
Tatsächlicher Zeitbedarf	Geplanter Zeitbedarf	Beschreibung der Arbeit	Bemerkungen, Probleme, genutzte Hilfestellungen
0.3h	0.5h	Einrichtung Slack und Dropbox.	Da ich auf meinem Dropbox Konto kein Speicherplatz mehr zur Verfügung hatte, habe ich ein neues Konto hinzugefügt.
0.5h	0.5h	Organisatorisches wie z.B. Sprint Meeting.	Wie immer haben wir am Anfang der Lektion das Sprint Meeting gehalten. Herr Walter hat uns auch noch Inputs zum Ablauf und den Abgaben geliefert.
2.1h	1.9h	An den Dokumenten arbeiten.	De Grössten Teil arbeitete ich an den Dokumenten. Das Arbeitsjournal stellte ich noch fertig, danach setzte ich mich wieder an den Realisierungsbericht.
Reflexion des Arbeitstages Heute hatten wir noch einmal viel zu tun. Es war der letzte Tag der Realisierungsphase. Gemeinsam mit den anderen vom Team arbeitete ich am Realisierungsbericht. Ausserdem habe ich bei unserem Projektplan Anpassungen vorgenommen. Ich hoffe das wir nicht mit der Abgabe in einen riesen Stress geraten und das Projekt «flöten» geht. Auch spannend war, dass wir herausgefunden haben, dass man ein Word Dokument im online Editor zeitgleich mit mehreren Personen bearbeiten kann. (In Dropbox öffnen.)			
Pendenzen für den nächsten Arbeitstag Da die Realisierungsphase soweit abgeschlossen ist, geht es in einem nächsten Schritt um die Einführung. Zudem bereiten wir unsere Präsentation vor.			

Teil 2

Situationsanalyse

Ausgangslage

In der App geht es darum, dass der Benutzer seine Schulnoten verwalten kann. Noten können eingetragen, bearbeitet und gelöscht werden. Darüber hinaus hat der Benutzer all seine Noten immer komfortabel im Überblick. Heute gibt es bereits Lösungen, die eine Verwaltung von Noten ermöglichen. Keine bestehende Lösung ist wirklich zufriedenstellend. Einige haben gute Benutzeroberflächen, sind aber von den Funktionen her nicht brauchbar. Andere haben viele Möglichkeiten, jedoch sind diese nicht benutzerfreundlich. Bei der Post CH AG gibt es ebenfalls eine bestehende Lösung, die aber Verbesserungspotential bietet. Die Problemstellung bei der neuen Lösung ist es, die Applikation den Wünschen und Bedürfnissen der Benutzerpalette so anzupassen, dass der Grossteil der User zufrieden ist. Sei es das User Interface oder der Ablauf innerhalb des Programms. Die Applikation soll stets verfügbar sein, auch unterwegs auf dem Smartphone. Das Programm soll alle nötigen Funktionen beinhalten, dabei aber die Benutzerfreundlichkeit beibehalten.

Stärken

Tabelle 7 Stärken der bestehenden Lösung

Nr.	Beschreibung
01	Die Schulnoten und Fächer sind in Semester unterteilt.
02	Noten von einem Fach können mit einer Gewichtung, Datum, Begründung und einem Typ gespeichert werden.
03	Die Noten können in einem Dokument exportiert werden.
04	Die Noten können nach Datum, Typ und Fach sortiert werden.

Schwächen*Tabelle 8 Schwächen der bestehenden Lösung*

Nr.	Beschreibung	Beurteilung	Ursache
S-01	Unübersichtliche Darstellung der Noten bei vielen Einträgen	Optimierungspotential durch Unterteilung aller Noten in die verschiedenen Fächer	Fehlende Unterteilung in einzelne Fächer in der Anwendung
S-02	Es können keine eigenen Fächer hinzugefügt werden	Optimierungspotential, wenn ein Fach nicht besteht und sonst jedes Mal ein Antrag an den Systemadministrator gemacht werden muss	Fehlende Funktionalität ein Fach hinzuzufügen in der Anwendung
S-03	Die Anwendung kann nur im Intranet der Post CH AG über einen Desktop PC oder Laptop erreicht werden	Hohes Optimierungspotential, wenn die Möglichkeit besteht die Applikation extern und über das Handy zu erreichen.	Fehlende Möglichkeit die Applikation ausserhalb des Post Netzes zu verwenden
S-04	Es ist nicht möglich das Prüfungsdokument hochzuladen	Es könnte durchaus nützlich sein, wenn man ein Bild oder eine PDF Datei des Prüfungsdokumentes bei der jeweiligen Note anhängen könnte.	Fehlende Möglichkeit eine Datei einer Note anzuhängen
S-05	Das Design der Applikation ist veraltet und somit macht es nicht Lust die Applikation zu nutzen.	Für den Benutzer wäre es sicherlich angenehm eine Applikation mit ansprechendem und modernen Design zu nutzen.	Die Applikation ist nicht mehr die neuste und somit wäre eine Revision angebracht.

Systemziele

Tabelle 9 Ziele

Nr.	Kategorie	Beschreibung	Messgrösse	Schwachpunkte	Priorität
Z-01	Grundfunktion	Der Benutzer kann eigene Fächer erstellen.	Ein Formular zur Erstellung eines Faches steht zur Verfügung.	S-02	Muss
Z-02	Grundfunktion	Das Layout ist ansprechend und funktional zu gebrauchen.	Es wird ein Material Design verwendet und der Aufbau wird so einfach wie möglich gehalten.	S-05	Muss
Z-03	Grundfunktion	Der Benutzer kann in wenigen Klicks eine neue Note hinzufügen.	Eine Note kann mit max. 10 Klicks erstellt werden.		Muss
Z-04	Grundfunktion	Der Benutzer hat die Möglichkeit nur die Noten eines bestimmten Faches anzeigen zu lassen.	Es gibt eine Ansicht, welche nur die Note des jeweiligen Faches anzeigt.	S-01	Muss
Z-05	Sicherheit	Es werden erforderliche Massnahmen getroffen um die Seite vor diversen Bedrohungen zu schützen.	Die Webapplikation wird vor SQL Injection, XSS und CSRF abgesichert.		Muss
Z-06	Qualität	Es wird mit einer Testinstallation auf einem Webserver bewiesen das die Applikation funktioniert.	Die Applikation kann fehlerfrei installiert werden und anschliessend die Grundfunktionen durchgetestet werden.		Muss
Z-07	Termin	Die Realisierung wird im zeitlichen Rahmen abgeschlossen.	Die Applikation ist nach 7 Wochen fertiggestellt.		Muss

Lösungsvorschläge

Variantenübersicht

Tabelle 10 Variantenübersicht

Variante	Bezeichnung
Variante a	Statische multi-page Applikation
Variante b	Dynamische single-page Applikation
Variante c	Native Android / iOS mobile Applikation
Variante d	Native Windows Desktop Applikation

Beschreibung der Varianten**Variante a**

Mit dieser Variante kann man eine Computer-freundliche Anwendung schaffen, die aber z.B. auf dem Smartphone nicht sehr praktisch ist. Ein Vorteil ist, dass das gesamte Entwicklerteam bereits Erfahrung mit solchen Applikationen hat. Die Möglichkeiten sind etwas eingeschränkt, weil es nicht sehr einfach ist von einer solchen Anwendung eine Smartphone-Applikation zu erstellen. Jedoch ist die Anwendung über das Internet erreichbar und somit grundsätzlich auf allen Geräten nutzbar.

Variante b

Diese Variante ermöglicht es das Programm auf Computern und Smartphones zu nutzen. Es handelt sich dabei um eine Webanwendung, welche auf modernen Technologien aufbaut. Das Design kann sehr ansprechend gestaltet werden und bei Bedarf ist es später möglich, eine Smartphone-Applikation daraus generieren zu lassen. Generell ist die Anwendung jedoch über das Internet erreichbar und somit von allen Geräten aus nutzbar.

Variante c

Bei dieser Variante wird eine Smartphone App entwickelt, welche man anschliessend aus dem App Store (Play Store) herunterladen könnte. Der Nachteil dieser Variante ist, dass die Anwendung anschliessend nur auf Smartphones verwendet werden kann und nicht auf dem Computer. Somit werden deutlich weniger Geräte unterstützt, als bei einer Webapplikation. Jedoch ist eine solche Applikation meistens performanter, als eine Internet-Applikation. Die Möglichkeiten sind jedoch etwas eingeschränkt, da man wie bereits erwähnt, nur eine Smartphone App erstellt.

Variante d

Diese Variante bietet dem Nutzer seine Noten per Computerprogramm zu bearbeiten. Der Vorteil dieser Möglichkeit ist, dass die Applikation unter Umständen viel flüssiger läuft, als eine Webanwendung. Jedoch ist hier das Problem, dass die Applikation nur auf einem Computer funktioniert und somit nicht auf dem Smartphone genutzt werden kann. Die Möglichkeiten sind aber begrenzt, da man eine Computer-App erstellt.

Bewertung der Varianten (Tabelle)

Variante a)

Tabelle 11 Variante a

Kriterium	Gewicht	Punkte	Total	
Abdeckung der Anforderungen	5	8	40	
Realisierbarkeit, Risiken	5	9	45	
Kosten	3	9	27	
Sicherheit	4	8	32	
Möglichkeiten	4	7	28	
Gesamtbeurteilung			172	

Beurteilungen: 1-10, Gewichtung 1-5

Variante b)

Tabelle 12 Variante b

Kriterium	Gewicht	Punkte	Total	
Abdeckung der Anforderungen	5	10	50	
Realisierbarkeit, Risiken	5	8	40	
Kosten	3	9	27	
Sicherheit	4	9	36	
Möglichkeiten	4	9	36	Es könnte leicht noch weiter in eine hybrid mobile Applikation umgewandelt werden.
Gesamtbeurteilung			189	

Beurteilungen: 1-10, Gewichtung 1-5

Variante c)*Tabelle 13 Variante c*

Kriterium	Gewicht	Punkte	Total	
Abdeckung der Anforderungen	5	6	30	Keine Möglichkeit die Applikation über den Computer zu öffnen. Auch für andere Smartphone Betriebssysteme nicht nutzbar.
Realisierbarkeit, Risiken	5	8	40	
Kosten	3	9	27	
Sicherheit	4	9	36	
Möglichkeiten	4	6	24	
Gesamtbeurteilung			157	

Beurteilungen: 1-10, Gewichtung 1-5

Variante d)*Tabelle 14 Variante d*

Kriterium	Gewicht	Punkte	Total	
Abdeckung der Anforderungen	5	6	30	Nur mit Microsoft Geräten nutzbar.
Realisierbarkeit, Risiken	5	8	40	
Kosten	3	9	27	
Sicherheit	4	9	36	
Möglichkeiten	4	6	24	
Gesamtbeurteilung			157	

Beurteilungen: 1-10, Gewichtung 1-5

Lösungsbeschreibung

Es wurde die Lösungsvariante b gewählt. Dies aus dem Grund, weil es dadurch möglich ist die Applikation auf allen nötigen Geräten zur Verfügung zu stellen. Mit der gewählten Technologie ist die Applikation webbasiert. Man nennt diese Art von Applikation auch «Single-page Application». Sie kann aber auch leicht zu einer hybriden mobilen Applikation umgewandelt werden, welche die Nutzererfahrung noch viel angenehmer macht.

Technologie

Angular [3] / Materialize [4]

Dies wird für die Frontend Programmierung verwendet. Angular ist ein Java Script Framework von Google, welches Clientseitig ausgeführt wird. Dazu verwenden wir das Materialize CSS Framework, welches die Google Material Design Standards zur Verfügung stellt.

Spring Data (REST / JPA)

Spring Data stellt dem Frontend die Daten des Backendes zur Verfügung. Dies erfolgt über JSON.

Spring Boot [2]

Ermöglicht einen einfachen Einstieg in die Welt von Spring. Weiter wird Spring Boot heute als Standard angeschaut und die meisten neuen Java Applikationen werden damit aufgebaut. Durch den Spring Initializr [1] ist es möglich die Spring-Grundkonfiguration mit der Auswahl der gewünschten Pakete zu generieren. Unter anderem werden auch Pakete wie Spring Security in Einsatz kommen.

Hibernate [5]

Ermöglicht Objektrelationales Mapping und ermöglicht somit Programmierern eine objektorientierte Sicht auf Tabellen und Beziehungen in relationalen Datenbank Management Systemen.

MySQL [6]

Eine viel verwendete, relationale Datenbank.

System-Skizze

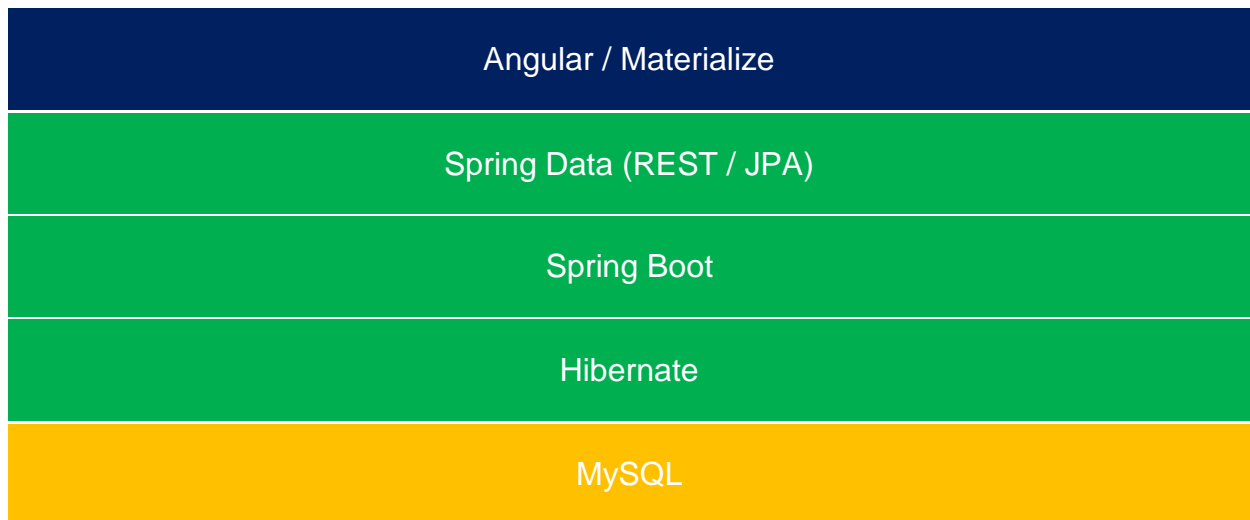


Abbildung 1 System-Skizze

Blau: Frontend, **Grün:** Backend, **Gelb:** Persistenz

Abgedeckte Anforderungen

Tabelle 15 Abgedeckte Anforderungen mit der Lösungsbeschreibung

Nr.	Anforderungen	Variante b
1	Ich als Benutzer kann einen neuen Account erstellen.	Ja
2	Ich als Benutzer kann mich mit meinem Account anmelden.	Ja
3	Ich als Benutzer kann Noten erfassen.	Ja
4	Ich als Benutzer kann ein Fach hinzufügen.	Ja
5	Ich als Benutzer kann meinem Account einen Beruf hinzufügen.	Ja

Systemarchitektur

Systemdesign

Struktur des Systemdesigns

Backend:

Das Backend Diagramm zeigt die wichtigsten Teile unserer Backend Applikation. Auch hier haben wir uns auf das wesentliche reduziert um die Übersichtlichkeit zu gewährleisten. Weiter konnten die verschiedenen Verbindungen nicht gezogen werden, weil die Diagramme aufgeteilt werden müssen, zur besseren Lesbarkeit.

Model

Dieses Diagramm zeigt die Models des Backendes in Gradus. Diese Java Models werden mit Hibernate gemappt und sind somit mit der Persistenzschicht, also der DB (siehe ERD) verknüpft.

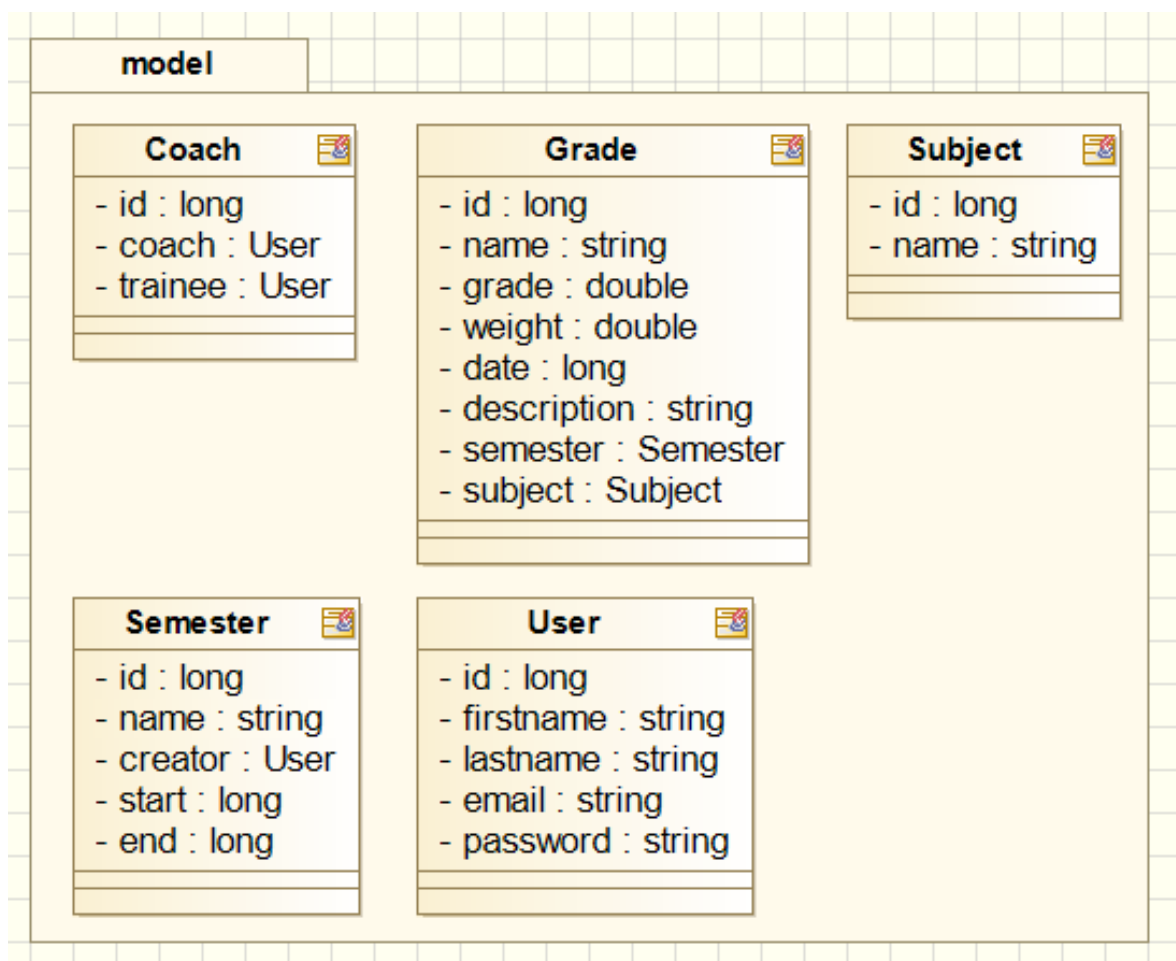


Abbildung 2 Backend Diagramm – Model

Controller

Das Backend Diagrammes des Controller Teiles ist auf der nächsten Seite sichtbar. Die Controller machen die Verbindung zwischen dem Backend und dem Frontend. Diese stellen über URLs Daten vom Backend dem Frontend zur Verfügung. Dies nennt man eine sogenannte REST Schnittstelle. Bei dieser Schnittstelle wird über eine URL Daten im JSON Format zur Verfügung gestellt.

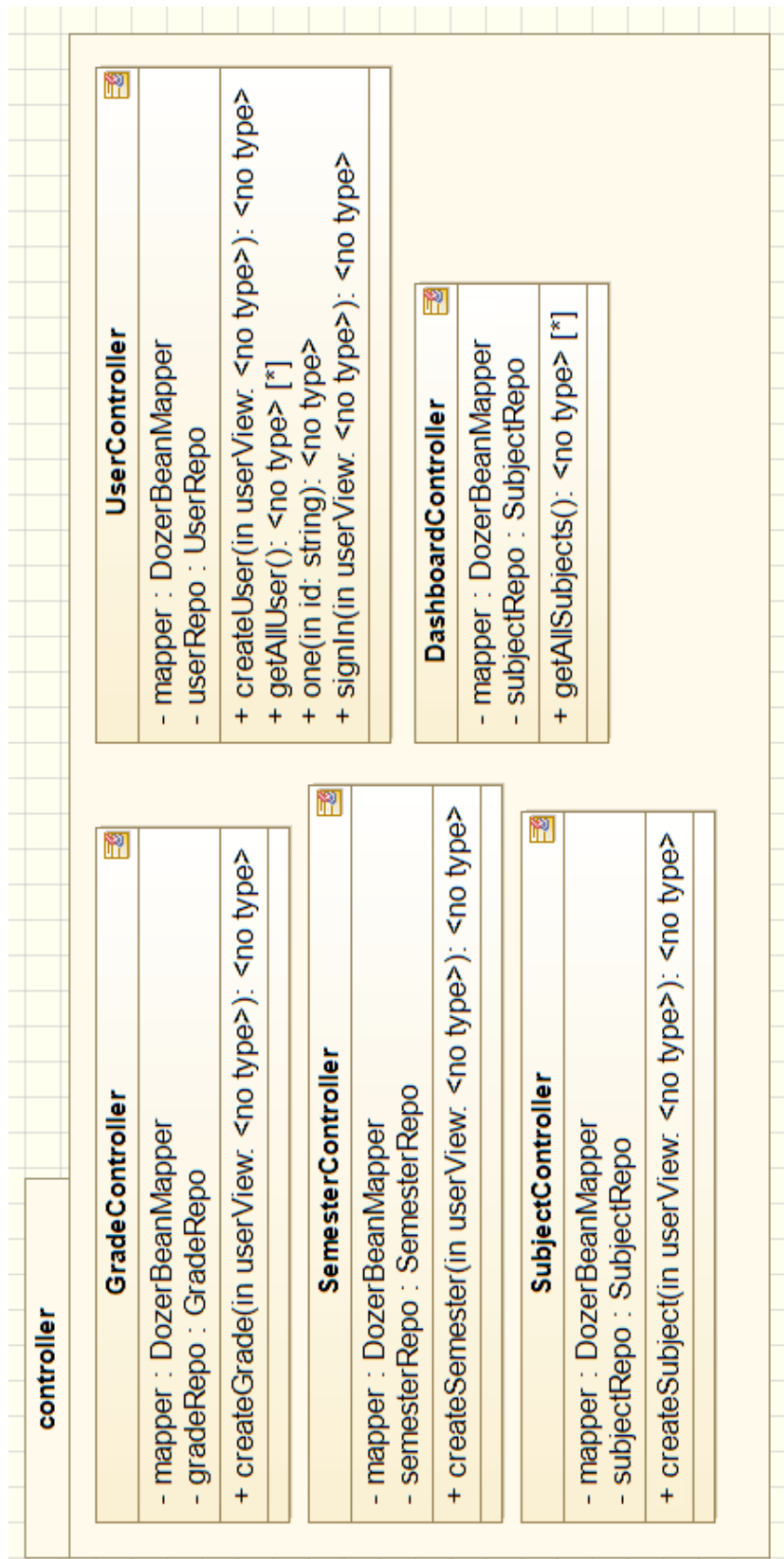


Abbildung 3 Backend Diagramm - Controller

Repo

Die Repos stellen die verschiedenen DB Abfragen zur Verfügung.

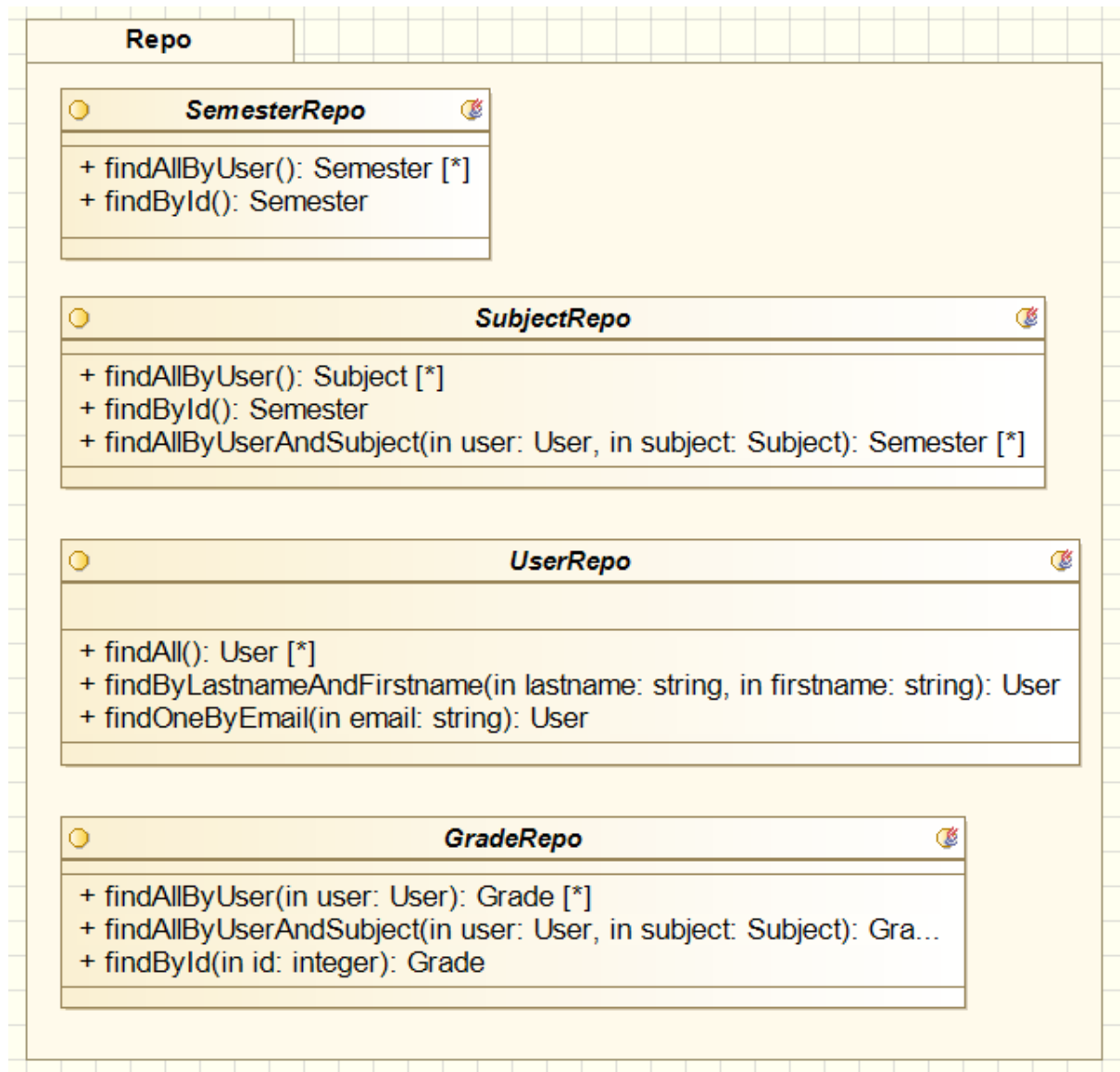


Abbildung 4 Backend Diagramm – Repo

Frontend:

Das Diagramm auf den nächsten 2 Seiten zeigt auf, wie die das Frontend der Applikation aufgebaut ist. Es wird nur das wichtigste aufgezeigt, damit die Verständlichkeit des Diagrammes nicht leidet. Es zeigt auf wie der Benutzer mit einem Request zugreift und grundsätzlich nur mit der index.html Datei kommuniziert. Dahinter arbeitet dann unsere Frontend Applikation und liefert anschliessend wieder alles an die index.html Datei. Dieser Teil der Applikation kommuniziert über die Services mit dem REST Backend. Dort werden Daten vom Backend geholt oder Daten ans Backend gesendet.

Frontend Diagramm Teil 1

In diesem Teil des Frontend Diagrammes sieht man die Bereiche Components und Models. Weiter auch noch den Backend Teil und den Ablauf mit dem Request des Benutzers.

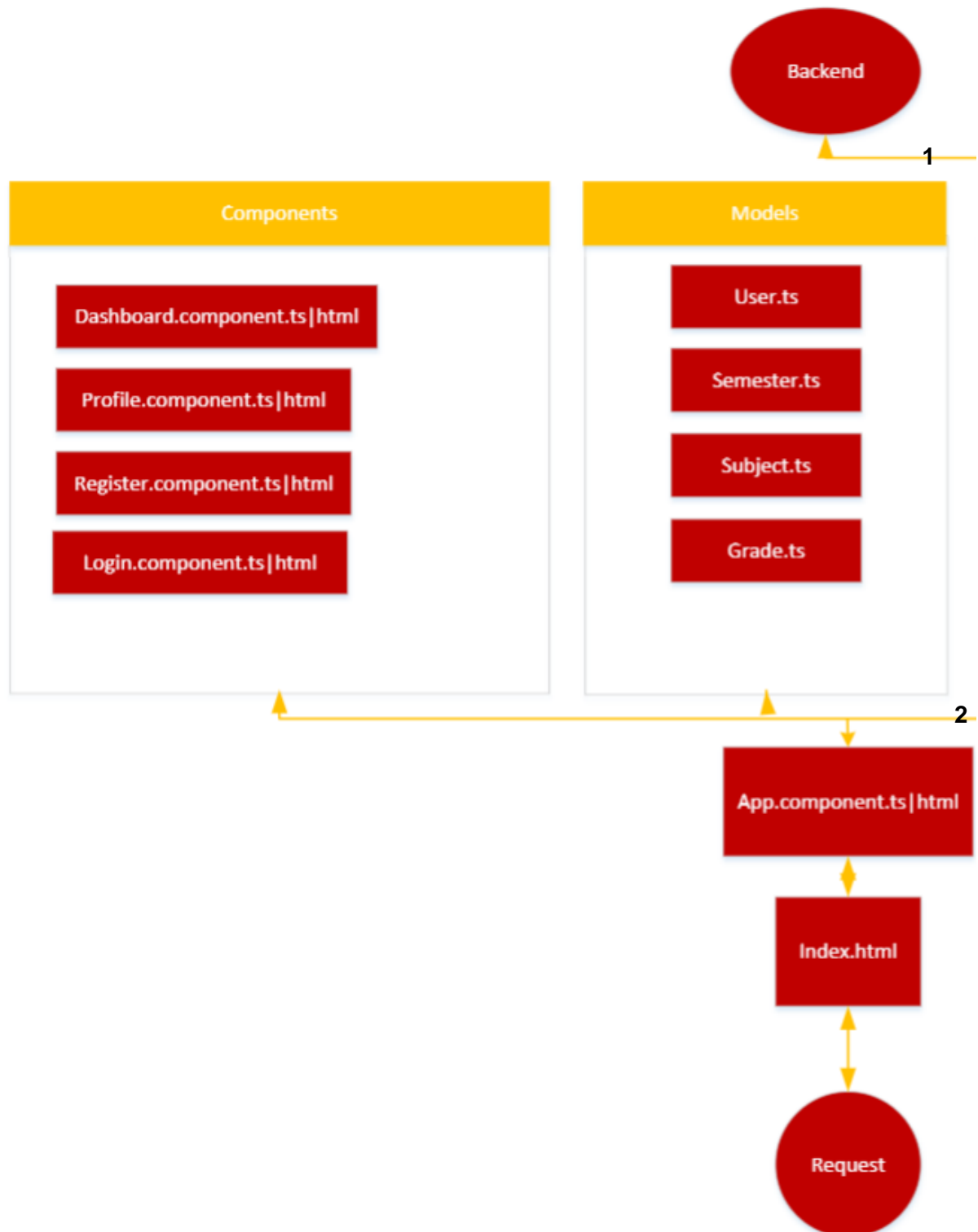


Abbildung 5 Frontend Diagramm Teil 1

Frontend Diagramm Teil 2

In diesem Teil des Frontend Diagrammes sieht man die Bereiche Services und Globals.

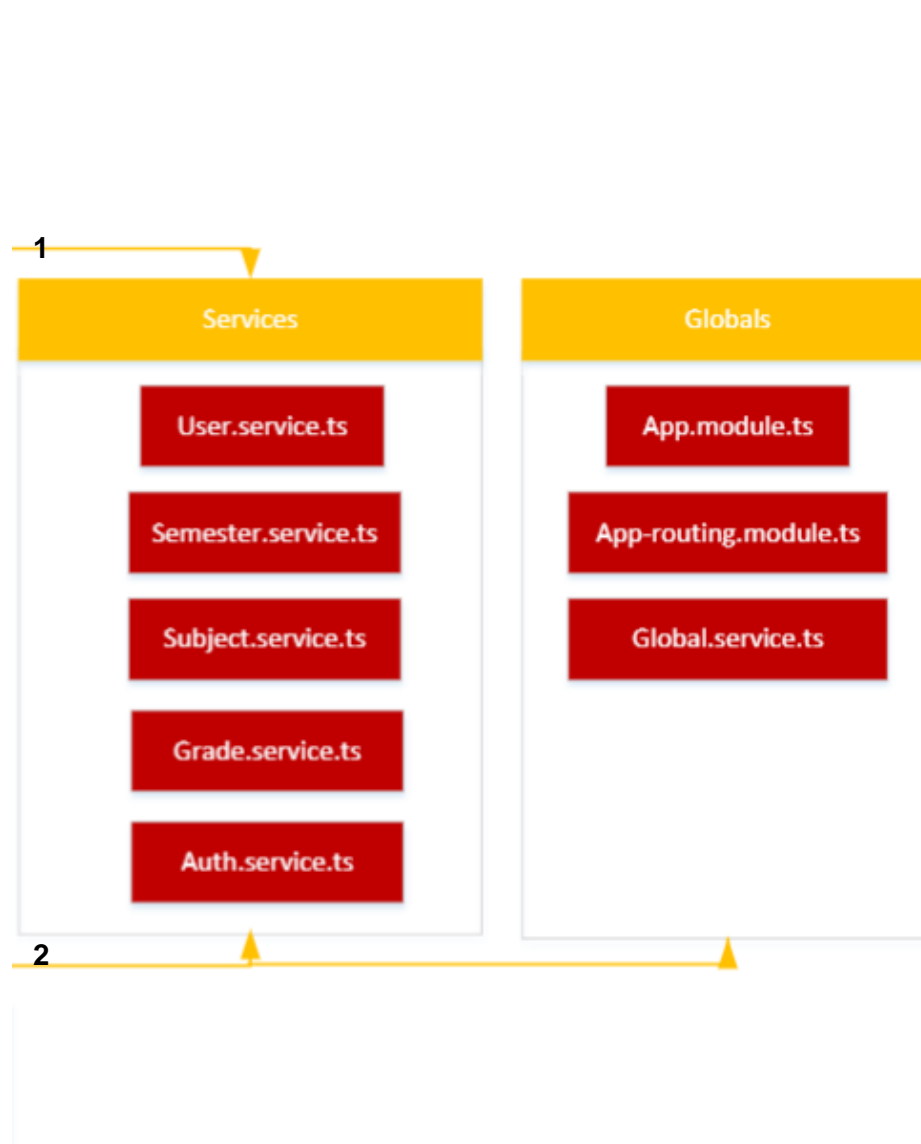


Abbildung 6 Frontend Diagramm Teil 2

*Beschreibung der Elemente***Backend:***Tabelle 16 Beschreibung der Elemente - Backend*

Element	Beschreibung
Model: <ul style="list-style-type: none">- Coach- Grade- Subject- User- Semester	Die Models halten die Daten der Applikation. Sie sind mit der DB Verknüpft über Hibernate. <ul style="list-style-type: none">- Coach ist damit ein Benutzer die Noten eines anderen einsehen kann.- Grade ist um Informationen zu den einzelnen Noten zu halten.- Subject ist für ein Fach.- User enthält alle Informationen über den Benutzer, jedoch wird das Passwort nicht über die REST Schnittstelle zur Verfügung gestellt.- Semester enthält Infos zum Semester.
Controller <ul style="list-style-type: none">- GradeController- SubjectController- SemesterController- UserController- DashboardController	Controller sind für die Verbindung zwischen Backend und dem Frontend zuständig. Sie kommunizieren per REST Schnittstelle. Es werden verschiedene URLs zur Verfügung gestellt, welche es dem Frontend ermöglichen Daten zu bekommen und Daten zu schicken. <ul style="list-style-type: none">- GradeController für Verwaltung der Noten.- SubjectController für die Verwaltung der Fächer.- SemesterController für die Verwaltung der Semester.- UserController für die Verwaltung des Benutzers.- DashboardController für die Verwaltung der gesamten Applikation. Sprich von Semester, Note und Fach.
Repo <ul style="list-style-type: none">- SemesterRepo- SubjectRepo- GradeRepo- UserRepo	Stellen verschiedene Abfragen zur Verfügung um Daten aus der DB zu holen oder Daten einzufügen. <ul style="list-style-type: none">- SemesterRepo für die Abfragen des gesamten Semesters oder allen Semestern.- SubjectRepo für die Abfragen von einem Fach oder von allen Fächern.- GradeRepo für die Abfragen von einem oder allen Noten eines Faches.- UserRepo für alles rund um den Benutzer selbst.

Frontend:

Zweck der wichtigsten Elemente:

Tabelle 17 Beschreibung der Elemente - Frontend

File	Beschreibung
frontend/src/ index.html	Startpunkt der Angular Applikation. Dort werden alle Stylesheets importiert, das Favicon und der Title gesetzt.
frontend/src/app/ app.module.ts	Dieses File importiert Module die in der gesamten Applikation verwendet werden. Weiter bootstrap dieses File die AppComponent.
frontend/src/app/ app-routing.module.ts	Dieses File kümmert sich um das Routing (Navigationsmöglichkeiten) der Applikation.
frontend/src/app/ app.component.ts html	Beinhaltet Globale Services und das Grundkonstrukt der Seite.
frontend/src/app/ login/	Beinhaltet die Login-Komponente.
frontend/src/app/ register/	Beinhaltet die Registrierungs-Komponente.
frontend/src/app/ dashboard/	Beinhaltet die Hauptkomponente, wo die Noten verwaltet werden.
frontend/src/app/ profile/	Beinhaltet die Benutzerprofil Komponente.
frontend/src/app/shared/ service/	Beinhaltet alle Services die von den Komponenten genutzt werden können. Diese Services holen Daten vom Backend über die sogenannten REST Schnittstellen.

Schnittstellendefinitionen

Gradus verfügt über folgende Schnittstellen:

REST (Backend – Frontend)

Die verschiedenen Controller im Backend stellen über URLs Daten dem Frontend zur Verfügung. Die Daten werden im JSON Format bereitgestellt. Heute ist dies eine sehr verbreitete Methode um das Frontend und das Backend vollständig trennen zu können und so bessere Applikationen schreiben zu können. Dies funktioniert wie folgt:

Der Controller im Spring Backend mappt eine Funktion, z.B. `getAllSubjects()` und stellt diese dann über die URL „`/webresources/subject`“ zur Verfügung. Dies erfolgt in dem Fall über eine GET Abfrage, würde aber auch mit POST funktionieren.

```
@RequestMapping(value = "/webresources/subject", method = RequestMethod.GET)
```

Abbildung 7 Backend Mapping

Anschliessend kann das Frontend über die passende URL auf die nötigen Ressourcen zugreifen. Danach kann man in einer Frontend Komponente auch einfach `getSubjects()` aufrufen und bekommt die Daten vom Backend zugestellt.

```
private baseUrl = 'http://localhost:8080/webresources/subject'; // URL to web api

getSubjects() {
    return this.http.get(this.baseUrl)
        .map(res => <Subject[]> res.json())
        .catch(this.handleError);
}
```

Abbildung 8 Frontend Mapping Zugriff

Hibernate (Backend – DB)

In einer sogenannten properties Datei werden die nötigen Informationen zur Datenbank gespeichert, damit die Applikation mit der DB operieren kann. Dies beinhaltet unter anderem die URL der bereitgestellten DB, der Name und das Passwort des Benutzers und ein paar nötige Daten zu Hibernate selbst.

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/gradus
spring.datasource.username=gradus
spring.datasource.password=gradus
spring.datasource.testOnBorrow=true
spring.datasource.validationQuery=SELECT 1
spring.jpa.hibernate.naming_strategy=org.hibernate.cfg.EJB3NamingStrategy
logging.level.org.hibernate=INFO
spring.jpa.hibernate.ddl-auto=update
```

Abbildung 9 Hibernate Config

Anschliessend werden die Model Klassen mit Hibernate gemappt und somit weiss die Applikation automatisch was sie machen muss, wenn eine Änderung getätigt wird. Im Bild sieht man wie eine Java Klasse mit einer DB Table gemappt wurde.

```
@Entity
@Table (name = "GRAD_user")
public class User {
```

Abbildung 10 Hibernate Mapping

Datenmodel

ERD

Im Diagramm auf der nächsten Seite sieht man das ERD von der Applikation Gradus. Es handelt sich dabei um eine MySQL Datenbank. Da die Applikation aber mit Hibernate arbeitet, wäre es Problem los möglich eine andere Datenbank, wie z.B. Oracle DB zu verwenden.

Beschreibung der Tabellen

Dies ist die Beschreibung der Tabellen des ERD Diagrammes auf der nächsten Seite.

Tabelle 18 ERD Beschreibung

ID	Tabelle	Beschreibung
T1	user	Beinhaltet alle Informationen über einen Benutzer, damit er identifiziert werden kann.
T2	user_subject	Verbindet den Benutzer mit seinen Noten für das jeweilige Semester. Denn der Benutzer kann für jedes Semester unterschiedliche Fächer haben.
T3	semester	Ermöglicht es Semester zu erstellen, mit einem Namen und einer Laufzeit.
T4	subject	Ein Fach von der Schule im Verbund mit der Schule bei dem das Fach besucht wird.
T5	institution	Beinhaltet die verschiedenen Bildungseinrichtungen.
T6	grade	Beinhaltet die einzelnen Noten mit ihrem Gewicht, Probenname, Probendatum, Gewichtung, Beschreibung und dem dazugehörigen Fach.
T7	coach	Die Tabelle Coach speichert, wenn ein User ein Coach von einem anderen User ist. Dies ermöglicht ihm dann, dass er einen Einblick in die Noten des Schülers haben kann.

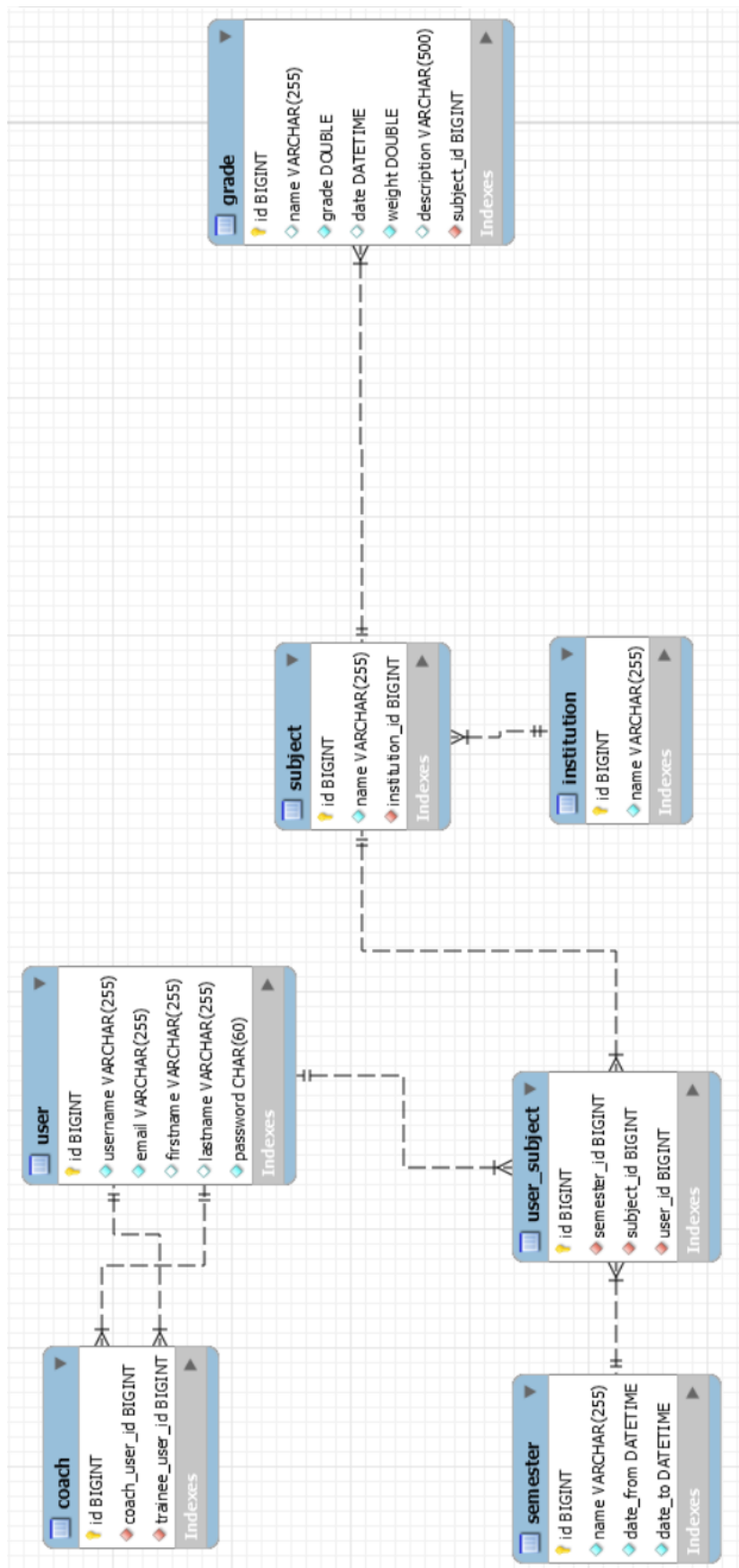


Abbildung 11 ERD

Sicherheit*Informationssicherheit**Tabelle 19 Informationssicherheit*

Beschreibung:	Gefährdungen:	Massnahmen:
Organisatorische und technische Massnahmen zur Sicherstellung von Verfügbarkeit und Authentizität der Daten.	<ul style="list-style-type: none">- Löschen von Daten von anderen Benutzern- Ausfall des Servers	<ul style="list-style-type: none">- Berechtigungskonzept implementiert und nur den Benutzern die die Noten gehören können diese auch bearbeiten.- Die Datenbank kann exportiert werden um den Schaden von Verlust von Daten möglichst gering zu halten. Dies sollte der Administrator in einem Cron-Script wöchentlich durchführen.

*Datenschutz**Tabelle 20 Datenschutz*

Beschreibung:	Schützenswerte Daten:	Massnahmen:
Schutz sensibler Daten vor unbefugtem Zugriff und vor missbräuchlicher Verwendung.	<ul style="list-style-type: none">- Benutzerinformationen- Notendaten- Source-Code	<ul style="list-style-type: none">- Die Benutzerdaten sind nur für den jeweiligen Benutzer einsehbar. Weiter hat auch der Coach Zugriff. Alle anderen sehen diese nicht.- Die Notendaten können ebenfalls nur vom Benutzer selbst oder vom Coach angesehen werden.- Der Source-Code wird in einem externen Repository gesichert, damit er nicht verloren gehen kann. Weiter ist er mit einer Lizenz versehen-

Testkonzept und Testspezifikationen

Systemtest - Testspezifikation

Kritikalität der Funktionseinheit

Die Tests werden mit der Stufe «hoch» oder «niedrig» eingestuft.
Ist ein Testfall positiv durchlaufen, so gilt sein Ausmass als «niedrig».
Ist ein Testfall negativ eingestuft, so gilt sein Ausmass als «hoch».

Testanforderungen

Die Tests sind mit fehlerfreien Werten durchzuführen.
Die Tests sind unter Normalbedingungen auszuführen.

Testverfahren

Das Testing braucht keine spezielle Vorbereitung. Die Schritte, so wie die erwarteten Resultate sind untenstehend aufgeführt. Die Daten werden während des Testdurchganges vom Tester erstellt.

Testkriterien

Abdeckungsgrad:

Alle Tests sind gemäss Testprotokoll durchzuführen. Es wird direkt mit der Applikation getestet.

Zum Testen werden Äquivalenzwerte verwendet(Stichprobenartig). Alle funktionalen Anforderungen (UserStories) müssen abgedeckt sein.

Checklisten:

Als Checkliste gilt das Testkonzept.

Endkriterien:

Alle Testfälle sind erfolgreich durchgeführt worden. Die Testperson hat das Testprotokoll korrekt ausgefüllt und unterzeichnet.

Testfälle

Tabelle 21 Testfälle

Testfall	User Story
Account erstellen	Ich als Benutzer kann einen neuen Account erstellen.
Anmeldung	Ich als Benutzer kann mich mit meinem Account anmelden.
Noten Erfassen	Ich als Benutzer kann Noten erfassen.
Fach hinzufügen	Ich als Benutzer kann ein Fach hinzufügen.
Note bearbeiten	Ich als Benutzer kann falsche Noten bearbeiten.
Fach bearbeiten	Ich als Benutzer kann einen Fachnamen ändern.
Benutzer bearbeiten	Ich als Benutzer kann meine Profildaten bearbeiten.
Note löschen	Ich als Benutzer kann eine unnötige Note löschen.
Fach löschen	Ich als Benutzer kann ein Fach löschen.
Benutzer löschen	Ich als Benutzer kann meinen Account löschen.
Semester hinzufügen	Ich als Benutzer kann ein Semester hinzufügen.
Semester Bearbeiten	Ich als Benutzer kann ein Semester Bearbeiten.
Semester löschen	Ich als Benutzer kann ein Semester löschen.

Einführungskonzept

Folgendes wird beim Projekt Gradus in der Einführung passieren.

Einführungsplan

Einführung

- Für die Überführung in den produktiven Betrieb ist eine Überführung von der Entwicklungsumgebung in die Produktion erforderlich. Die Applikation wird auf einen Server geladen, welcher dann über das Internet verfügbar ist. Diese Instanz der Applikation wird für alle Personen von überall zur Verfügung stehen.
- Dadurch das wir diese Applikation gratis und für alle zur Verfügung stellen, können wir uns eine zusätzliche Pilotphase sparen und direkt auf dem produktiven System durchführen.

Selbstverständlich muss aber zuerst garantiert werden, dass die Daten sicher sind und die Benutzer informiert werden, dass das Produkt noch nicht 100% zuverlässig ist.

Migration

- Falls diese Online-Version von Gradus gut ankommt und eventuell noch einige Erweiterungen an der Applikation vorgenommen wurden, wird auch in Betracht gezogen das Notenerfassungssystem der Post auf unser System zu migrieren. Dies wäre jedoch ein riesiger Aufwand und müsste daher nochmals gut durchgeplant werden.

Schulung

- Das System ist grundsätzlich so aufgebaut, dass es selbsterklärend ist und möglichst einfach.
- Weiter haben der grösste Teil, der möglicherweise zukünftigen Kunden bereits Erfahrungen mit einem Notenerfassungstool und daher kennen diese die wichtigsten Grundfunktionen bereits, welche auch bei Gradus vorhanden sind.
- Trotzdem wird in Betracht gezogen noch eine Einführung durch ein kleines Video-Tutorial zu geben.
- Weiter steht auch noch das Benutzerhandbuch zur Verfügung.

Grober Ablauf der Inbetriebnahme:

1. Schreiben eines Scripts zur einfachen Installation auf einem neuen System
2. Aufsetzen des Webserver
3. Überführung des Systems von der Entwicklungsumgebung auf den neuen Webserver
4. Verknüpfen einer Domain mit dem Webserver
5. Freigabe des Systems
6. Möglicherweise Erstellen von Schulungsvideos
7. Nach längerer Testphase der Online-Version möglicherweise Migration vom Post Notenerfassungssystem auf Gradus

Migrationsplan

Migrationsverfahren

Grundsätzlich ist zu Beginn keine Datenübernahme nötig, weil wir das System frisch aufsetzen.

Es ist jedoch nötig, wenn wir das System später bei PostFinance einsetzen, denn dort hat es bereits eine grosse Menge an Daten vom aktuellen Notenerfassungstool.

Dann müsste folgendes geschehen:

- DB Schemen so anpassen, dass eine konfliktlose Migration der Daten möglich ist.
- Anpassen der Frontend-Funktionen in Gradus, damit es alle vorherigen Möglichkeiten im neuen Produkt auch beinhaltet.
- Testen einer Migration von einem alten auf das neue Produkt auf einem Testsystem.
- Bei erfolgreicher Durchführung könnte man dann einen Termin abmachen, wo das aktuelle System möglichst kurz heruntergefahren wird und die Migration vom alten auf das neue Produkt erfolgt. Dieser Termin sollte an einem Tag sein, an dem keine Lehrlinge bei der Arbeit sind, somit am Wochenende, damit die Downtime weniger ärgerlich ist.

Zeitplan für die Migration

Überführung auf den neuen Webserver

Meilensteine:

- Produktionsscript fertigstellen
- Webserver aufsetzen
- Deployen des Produktes auf dem neuen Webserver
- Domain einrichten um Zugriff von aussen zu ermöglichen
- Testen auf dem produktiven System
- Freigabe des Produktes für die Benutzer

Migration altes Post Notenerfassungstool auf Gradus

Meilensteine:

- DB von Gradus erweitern (mit den nötigen Tabellen vom alten System ergänzen)
- Frontend von Gradus erweitern, damit alle Möglichkeiten vom alten System vorhanden sind
- Testsystem einrichten für die Migration
- Migration auf Testsystem durchführen (von altem auf neues Notenerfassungstool)
- Richtige Migration durchführen

Organisation der Migration

Tabelle 22 Kontakte Migration

Personen	Kontakt
Entwicklerteam <ul style="list-style-type: none">- Mirio Eggmann- Manuel Bieri- Nicolas Brechbühler- Dario Menzel	mirio.eggmann@protonmail.ch
Beat Walter - Auftraggeber	beat.walter@iet-gibb.ch
Thomas Käser – Ansprechperson PostFinance	thomas.kaeser.1@postfinance.ch

Testprotokoll

Der Test wurde erfolgreich durchgeführt.

Folgend, steht hier das Testprotokoll aus dem *Realisierungsbericht*:

Tabelle 23 Testprotokoll

Testfall		Erfüllt	Bemerkung
1.	Account erstellen	<input checked="" type="checkbox"/>	
2.	Anmeldung	<input checked="" type="checkbox"/>	
3.	Noten Erfassen	<input checked="" type="checkbox"/>	
4.	Fach hinzufügen	<input checked="" type="checkbox"/>	
5.	Note bearbeiten	<input checked="" type="checkbox"/>	
6.	Fach bearbeiten	<input checked="" type="checkbox"/>	
7.	Benutzer bearbeiten	<input checked="" type="checkbox"/>	
8.	Note löschen	<input checked="" type="checkbox"/>	
9.	Fach löschen	<input checked="" type="checkbox"/>	
10.	Benutzer löschen	<input checked="" type="checkbox"/>	
11.	Semester hinzufügen	<input checked="" type="checkbox"/>	
12.	Semester Bearbeiten	<input checked="" type="checkbox"/>	
13.	Semester löschen	<input checked="" type="checkbox"/>	

Abnahme

Tabelle 24 Abnahme

Testdatum	13.12.2016
Tester	Dario Menzel
Gesamttestresultat	<input checked="" type="checkbox"/> Abgenommen <input type="checkbox"/> Abgenommen mit Nacharbeiten <input type="checkbox"/> Nicht abgenommen
Nacharbeiten	-
Unterschrift Lieferant	Entwicklerteam
Unterschrift Kunde	Beat Walter

Benutzerdokumentation / Anleitung

Benutzerhandbuch

Systemübersicht

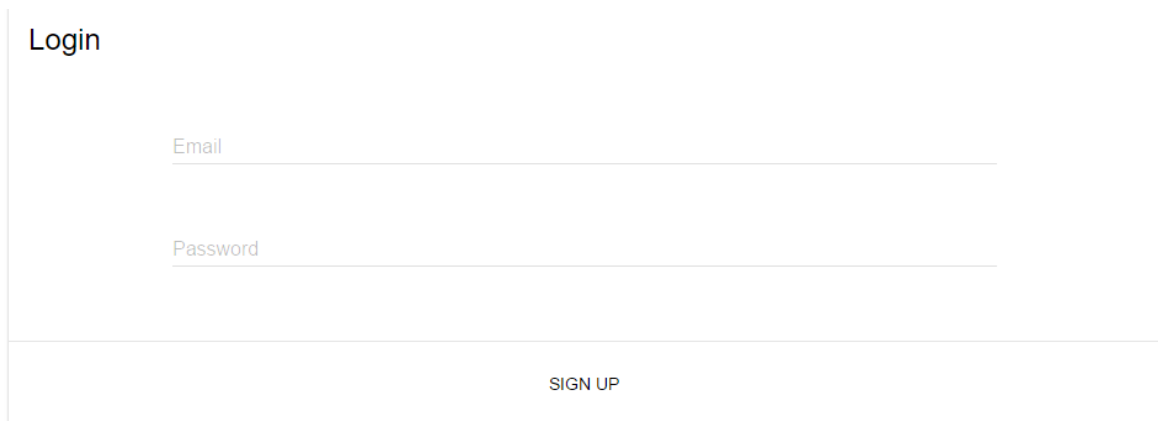
Ziel dieses Systems ist es, die momentane Applikation zur Notenerfassung der Post CH AG, durch unsere Web-Anwendung abzulösen. In Gradus geht es darum, dass der Benutzer seine Schulnoten verwalten kann. Noten können eingetragen, bearbeitet und gelöscht werden. Darüber hinaus hat der Benutzer all seine Noten, Semester und Gesamtschnitte immer komfortabel im Überblick. Die Applikation soll stets verfügbar sein, sogar unterwegs auf dem Smartphone. Jeder User erstellt für sich selbst einen Account mit E-Mail-Adresse und Benutzerpasswort, mit welchem er oder sie sich später über das Web einloggen kann und Zugriff auf den ganz persönlichen Notenerfassungsbereich erhält.

Anwenderfunktionalität

Sobald man das Interface der Anwendung im Webbrowser erreicht hat, kann sich die Benutzerin oder der Benutzer einen neuen Account erstellen oder sich mit bestehenden Benutzerdaten anmelden. Nach erfolgreicher Anmeldung landet man im Hauptbereich der Applikation. Es sind unter anderem Schaltflächen wie «Dashboard», «Profil» und «Logout» vorhanden. Darüber hinaus hat der User einen direkten Überblick über seine bereits eingetragenen Noten und die Möglichkeit, gleich neue Noten hinzuzufügen. Es können ebenfalls Einstellungen aufgerufen werden. Die Optionen welche zur Verfügung stehen in der Anwendung sind alle selbsterklärend und verfügen über eine grafisch passende Schaltfläche.

Login:

Falls sie noch keinen Account erstellt haben müssen Sie zuerst oben rechts in der Navigation auf Register klicken. Ansonsten können Sie sich mit Ihrer Email und Ihrem Passwort einloggen.



The image shows a login form with the following elements:

- A header labeled "Login".
- An input field labeled "Email".
- An input field labeled "Password".
- A button labeled "SIGN UP" at the bottom right.

Abbildung 12: Login-Seite

Registration:

Falls Sie nicht bereits einen Account erstellt haben, können Sie dies hier tun. Sie müssen nur Ihren Namen, Vornamen, Ihre Email und das gewünschte Passwort eingeben. Anschliessend können Sie sich damit bei Gradus registrieren.

Register

Firstname

Lastname

Email

Password

Repeat Password

SIGN UP

Abbildung 13: Registrations-Seite

Profil:

Auf dieser Seite sehen Sie Ihre Benutzerinformationen und über das Stift Symbol können Sie diese bearbeiten. Beim betätigen des Papierkorbes wird Ihr Benutzeraccount gelöscht.

Gradus

Dashboard

Profile

Logout

User information

firstname: Manuel

lastname: Bieri

email: bierimanu@gmail.com

Actions

Subjects

Mathematik

M946

Deutsch

M947

Englisch

M306

Abbildung 14: Profil-Seite

Note hinzufügen:

Hier können Sie neue Noten hinzufügen und speichern. Nebst den Noten selbst können auch noch Informationen wie ein Name, die Gewichtung der Note, sowie das Datum und sogar eine Beschreibung hinzugefügt werden. Durch Knopfdruck auf «ADD» werden Ihre Eingaben bestätigt. Beinahe dieselbe Ansicht bekommen Sie auch beim Noten bearbeiten, daher wird sie nicht nochmals dargestellt.

Add grade

Name

Grade

Weight

Date

Description

ADD

Abbildung 15: Seite zum Hinzufügen neuer Noten

Fach hinzufügen:

Auf dieser Seite können neue Fächer hinzugefügt werden. Dazu geben Sie bloss den Namen des Fachs, sowie den der Institution an. Durch einen abschliessenden Klick auf «ADD», wird das neue Fach mit Ihren Angaben gespeichert. Beinahe dieselbe Ansicht bekommen Sie auch beim Fach bearbeiten, daher wird sie nicht nochmals dargestellt.

Add subject

Name

Institution

ADD

Abbildung 16: Seite zum Hinzufügen neuer Fächer

Semester hinzufügen | bearbeiten:

Ein Semester können Sie hier hinzufügen, indem Sie dessen Namen eingeben und danach mit Knopfdruck auf «ADD» bestätigen. Beinahe dieselbe Ansicht bekommen Sie auch beim Semester bearbeiten, daher wird sie nicht nochmals dargestellt.

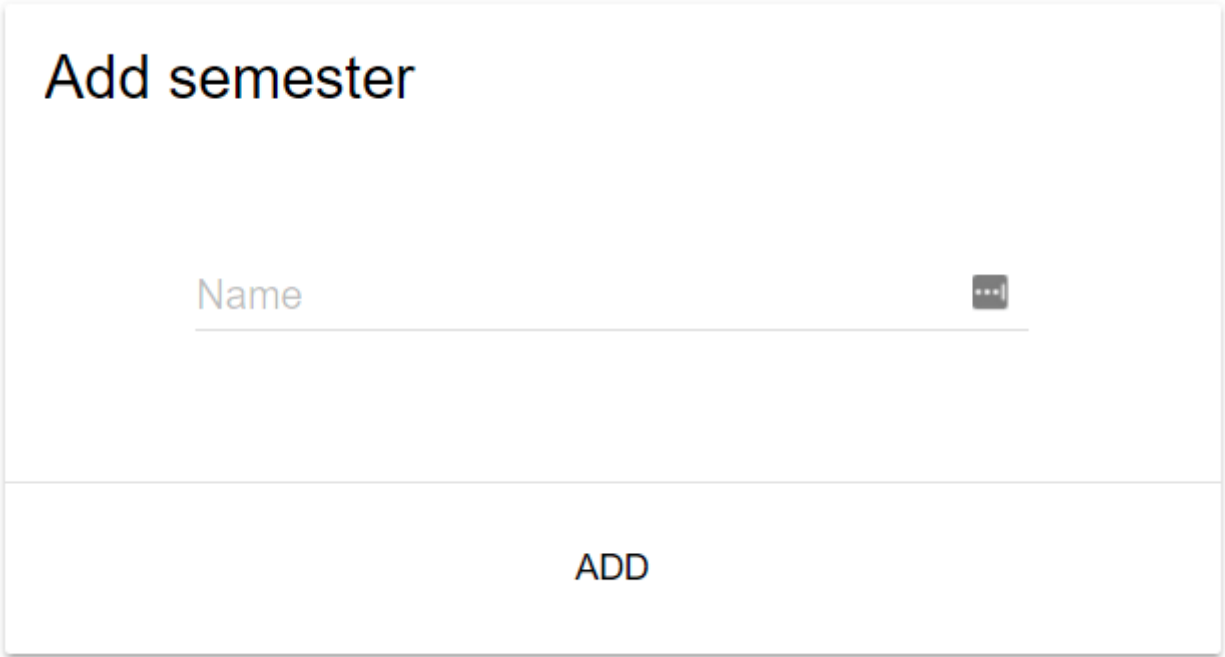
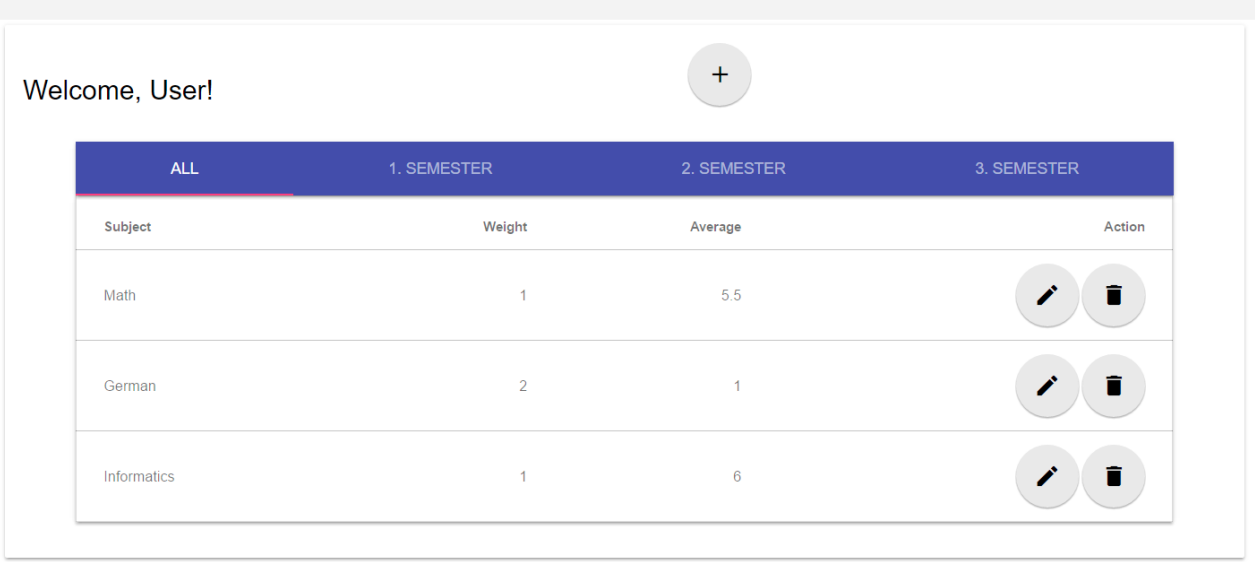


Abbildung 17: Seite zum Hinzufügen neuer Semester

Dashboard Übersicht:

Hier befinden Sie sich auf der Übersichtsseite Ihrer Noten. Zu jeder Note sieht man das Fach, das Semester und den Durchschnitt aller Noten eines Fachs. Hier stehen Ihnen wieder die Aktionen zum Bearbeiten oder Löschen zur Verfügung. Einen Eintrag bearbeiten können Sie, indem Sie am Ende einer Zeile auf das Stift-Symbol klicken. Wenn Sie eine Zeile löschen wollen, so klicken Sie einfach auf das Papierkorb-Symbol am Ende der entsprechenden Zeile.









ALL	1. SEMESTER	2. SEMESTER	3. SEMESTER
Subject	Weight	Average	Action
Math	1	5.5	 
German	2	1	 
Informatics	1	6	 

Abbildung 18: Dashboard Übersichts-Seite

Fach Übersicht:

Für die Fächer haben Sie jeweils eine separate Übersicht. Hier werden Ihnen alle Noten zum ausgewählten Fach angezeigt. Auch hier besteht wieder die Möglichkeit, Zeilen zu bearbeiten oder löschen. Einen Eintrag bearbeiten können Sie, indem Sie am Ende einer Zeile auf das Stift-Symbol klicken. Wenn Sie eine Zeile löschen wollen, so klicken Sie einfach auf das Papierkorb-Symbol am Ende der entsprechenden Zeile.

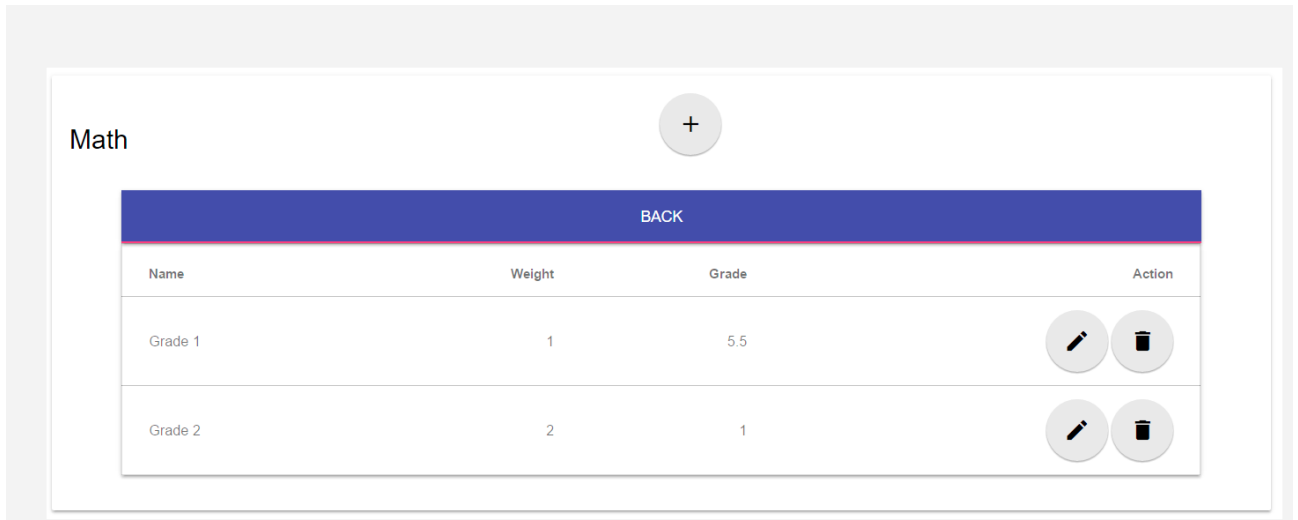


Abbildung 19: Fach Übersichts-Seite

Bei Fehler oder Fehlermeldung:

Aktualisieren Sie die Web-Applikation, indem Sie auf der Tastatur F5 betätigen oder im Webbrowser auf den - Knopf drücken (befindet sich meistens gleich rechts von der URL bar). Sollte dies nicht gereicht haben, versuchen Sie sich abzumelden und die Web-Applikation zu schliessen und sich danach erneut anzumelden. Sie können ebenfalls den Verlauf, Cookies und andere Website-Daten in den Browsereinstellungen löschen und es danach erneut probieren.

Integrations- und Installationshandbuch

Anwender

Um unsere Anwendung zu benutzen müssen zuerst folgende Kriterien eintreffen.

- Ein internetfähiges Gerät muss im Einsatz sein
- Ein Browser, wie Google Chrome muss installiert sein
- Es besteht eine Internetverbindung

Treffen zumindest mal diese Faktoren ein, ist es ein nächster Schritt Gradus zu installieren.

- Systeminstallation durchführen.

Zu einem späteren Zeitpunkt, anfangs 2017, wird es möglich sein Gradus über diese URL aufzurufen und zu nutzen. Dies wird ein Dienst sein, welcher jeder nutzen kann und somit nicht ein eigener Server für die Applikation betrieben werden muss: <https://iet-gibb-gradus.mirioeggmann.ch>

Wenn Gradus erfolgreich installiert wurde oder bereits anfangs 2017 ist und die Webseite online verfügbar ist, können sie sich nun anmelden oder falls sie noch kein Konto haben registrieren.

Systeminstallation

Benötigte Software:

Tabelle 25 Benötigte Software

Software	Beschreibung	Link
git	Benötigt es um dem Projekt einen Beitrag zu leisten und um immer die neuste Version herunterladen zu können. Weiter wird die Git Bash benötigt um die angular-cli zu bedienen.	https://git-scm.com/downloads
Java JDK 8	Java wird für das Spring Backend benötigt und muss somit installiert werden.	http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
NodeJS	Es wird die neuste NodeJS Version benötigt, damit Angular-cli funktioniert und die Projekt-Abhängigkeiten heruntergeladen werden können.	https://nodejs.org
Angular-cli	Diese wird benötigt um das Frontend zu starten. Es wäre theoretisch auch mit npm start möglich, jedoch ist es umständlicher.	https://github.com/angular/angular-cli
MySQL Server	Der MySQL Server wird benötigt um die Datenbank der Applikation zu betreiben.	https://dev.mysql.com/downloads/file/?id=466001

MySQL Workbench	Dies kann genutzt werden um einen Einblick in das MySQL Datenbanksystem zu bekommen.	http://dev.mysql.com/downloads/workbench/
IntelliJ IDEA	Mit dieser Entwicklungsumgebung wurde entwickelt und es wird empfohlen dies weiter zu führen.	https://www.jetbrains.com/idea/
Tomcat 8	Dies ist der Server auf dem die Java Spring Boot Applikation ausgeführt wird.	https://tomcat.apache.org/download-80.cgi

Installation

Datenbank

Voraussetzungen:

- MySQL Server installiert

Vorgehen:

- Im MySQL Server einloggen als root und ein neues DB Schema «gradus» erstellen.
- Einen neuen Benutzer anlegen mit dem Namen «gradus» und Passwort «gradus»
- Dem Benutzer «gradus» alle Rechte auf das Schema «gradus» geben.

Backend

Voraussetzungen:

- IntelliJ IDEA installiert
- Tomcat 8 wurde installiert
- Git installiert
- Java JDK 8 installiert

Vorgehen:

- Projekt Klonen, falls nicht bereits gemacht, mit der Git Bash: **git clone** <https://github.com/mirioeggmann/gradus.git>
- [Projekt konfigurieren, das es den Tomcat 8 Server benutzt](#)
- [Projekt mit IntelliJ starten.](#)

Frontend

Voraussetzungen:

- Git installiert
- NPM installiert
- Angular-cli installiert

Vorgehen:

- Projekt Klonen, falls nicht bereits gemacht, mit der Git Bash: **git clone** <https://github.com/mirioeggmann/gradus.git>
- Im Ordner /frontend **npm install** ausführen
- Frontend starten mit **ng serve**

Nun sollten sie in der Lage sein über localhost:4200 auf Gradus zuzugreifen.

Supporthandbuch

Gradus wird von keiner Support-Organisation unterstützt und somit bieten wir keinerlei Support für die Applikation. Wenn eine dringende Frage offensteht, können Sie mirio.eggmann@protonmail.ch kontaktieren.

Projekterfahrung

In diesem Projekt haben wir relativ viel Erfahrung gesammelt. Alle 4 von uns kannten einander bereits von unserem Arbeitgeber, der Post CH AG, und wir haben dementsprechend schon 2 Jahre miteinander verbracht. Somit waren wir optimistisch, dass wir schon wissen, wie jede Person arbeitet und wie wir zusammen vorwärtskommen. Dadurch haben wir uns entschieden viele neue Technologien zu verwenden, welche wir vorher noch nicht benutzt haben, um das Projekt anspruchsvoller zu machen und auch viele neue Erfahrungen zu sammeln. Es hat sich aber herausgestellt, dass wir viel zu wenig Zeit hatten, um uns noch in die neuen Technologien einzuarbeiten und dann effizient vorwärts zu kommen. Wir haben also in diesem Projekt unter anderem gelernt, dass man sich zuvor bereits besser in die nötigen Technologien einarbeiten muss und eventuell zusammen kleine Workshops macht, damit wirklich alle eine gewisse Grundbasis haben. Andererseits würde man vermutlich in einem richtigen Projekt nicht für total neue Technologien entscheiden. In diesem Projekt hatten wir das Problem, dass schlussendlich nur Manuel Bieri von Anfang an wusste, wie das meiste funktioniert. Mirio Eggmann hatte sich nur ganz wenig einarbeiten können, weil die Zeit schlicht zu kurz war. Dario Menzel und Nicolas Brechbühler konnten sich beinahe gar nicht einarbeiten und konnten schlussendlich beim Programmieren nicht wirklich mithelfen. Weiter haben wir auch zu wenig bedacht, dass die Pflicht Dokumente im Unterricht auch immer wieder viel Zeit rauben und somit beinahe keine Zeit mehr blieb, um im Unterricht zu Programmieren. Grundsätzlich hätte man fast das Programm zuhause schreiben müssen, jedoch hatten weder Manuel Bieri noch Mirio Eggmann Zeit dafür. Dies unter anderem auch, weil wir in den anderen Modulen ebenfalls noch Projekte hatten, die wir zuhause erledigen mussten.

Teil 3

Selber erstellte Listings und Skripte

Im Anhang A finden Sie kommentierten Sourcecode. In diesem Abschnitt erfolgt daher nur eine Auflistung der Filestruktur von unserem Projekt, um diese auch noch abzubilden.

Backend Filestruktur:

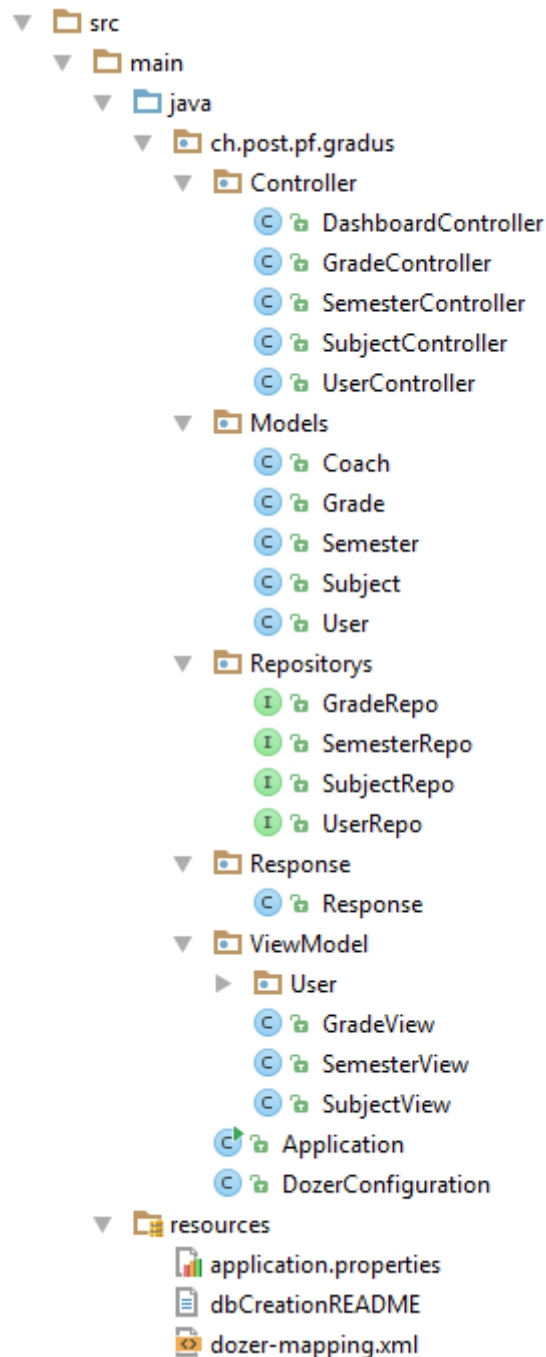


Abbildung 20 Backend Filestruktur

Frontend Filestruktur Part 1:

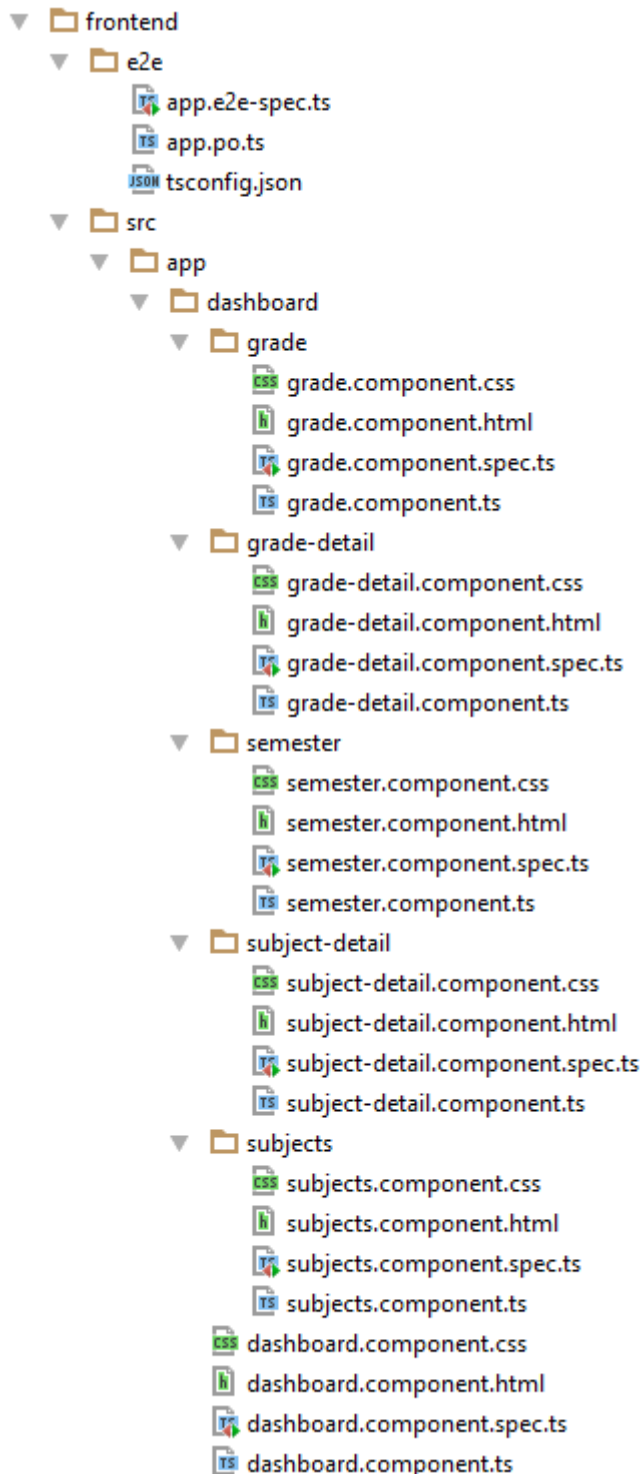


Abbildung 21 Frontend Part 1 Filestruktur

Frontend Filestruktur Part 2:

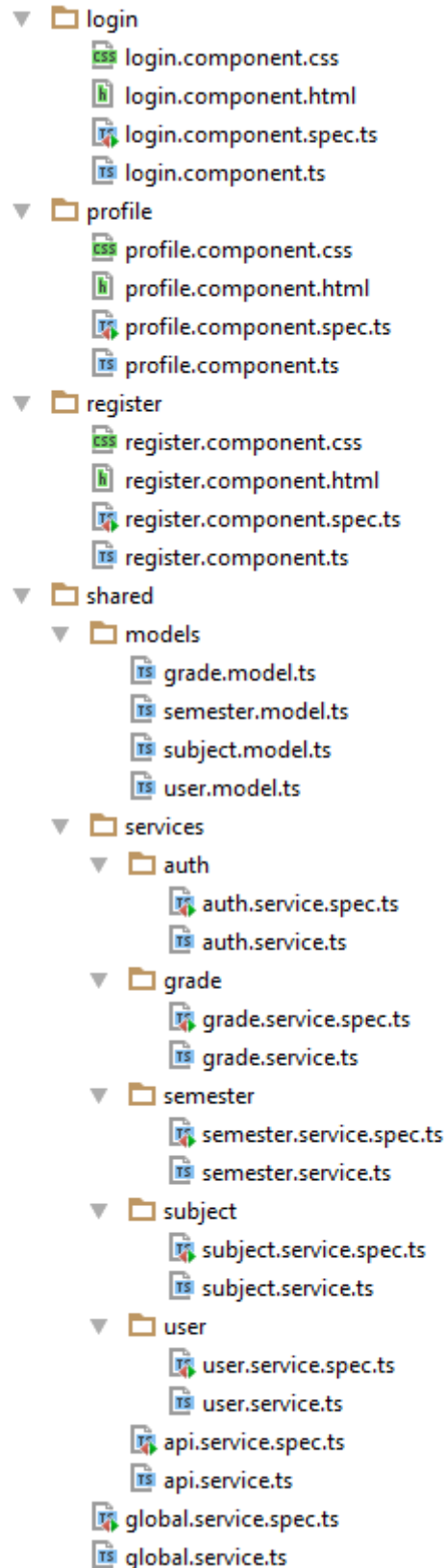


Abbildung 22 Frontend Part 2 Filestruktur

Frontend Filestruktur Part 3:

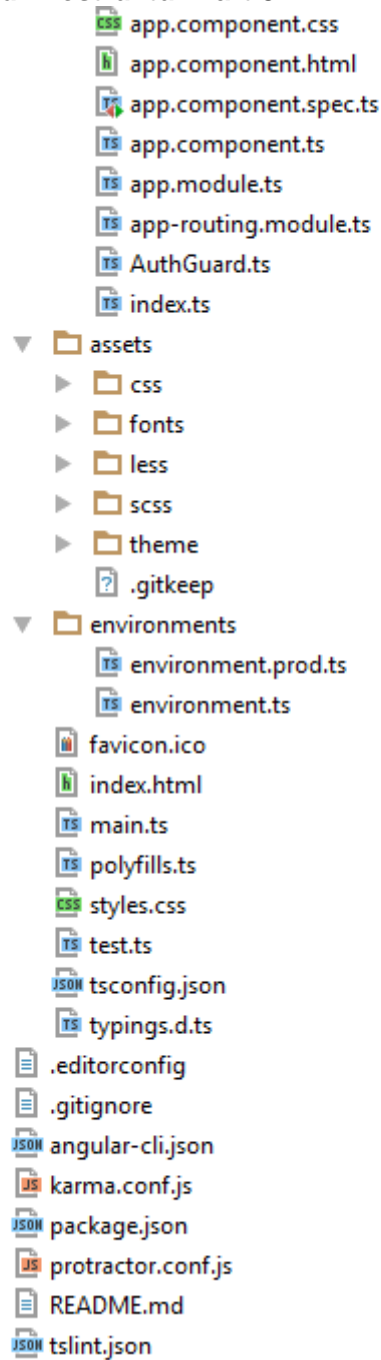


Abbildung 23 Frontend Part 3 Filestruktur

Literaturverzeichnis

Dokumentation Angular 2: <https://angular.io> (10.1.2017)

Dokumentation Spring: <https://spring.io> (10.1.2017)

Glossar

Tabelle 26 Glossar

Begriff / Abkürzung	Bedeutung
JDK	Java Developement Kit
Single-page	Eine moderne Webanwendung die dynamisch Inhalt lädt.
Multi-page	Eine Webanwendung die bei jeder Anfrage den gesamten Inhalt neu lädt.
GIBB	Gewerblich-Industrielle Berufsschule Bern
App	Abkürzung für Applikation / Anwendung.
PDF	«Portable Document Format» Plattformunabhängiges Datei Format
REST	«Representational State Transfer» Schnittstelle zwischen zwei Systemen
Play Store	Google Marktplatz um Applikationen herunterzuladen.
SQL	«Structured Query Language» Datenbanksprache
MDL	Material Design Lite

Anhang

A. Sourcecode

In diesem Abschnitt finden Sie kommentierten Sourcecode. Die einzelnen Bilder wurden bewusst nicht im Abbildungsverzeichnis festgehalten, weil es dadurch unübersichtlich werden würde und es auch wenig sinnvoll wäre dies in diesem Fall so festzuhalten. Den Sourcecode findet man auch auf Github, in einem Release, welcher aufweist, wann die letzte Änderung vorgenommen wurde.

Unter folgendem Link kann dies gefunden werden:

<https://github.com/mirioeggmann/gradus/releases/tag/v1.0.0>

Falls Sie wirklich interessiert sind und auch Änderungen an Gradus vornehmen wollen, sollten Sie sich den aktuellsten Code auf Github anschauen:

Link zum Repository: <https://github.com/mirioeggmann/gradus>

DashboardController.java

```
package ch.post.pf.gradus.Controller;

import ch.post.pf.gradus.Models.Subject;
import ch.post.pf.gradus.Models.User;
import ch.post.pf.gradus.Repositorys.SubjectRepo;
import org.dozer.DozerBeanMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

/**
 * This class provides the methodes for the frontend to load everything that is needed on of the dashboard.
 * @author manuel.bieri
 * @version 1.0.0 - 02.01.2017
 */
@RestController
@CrossOrigin
public class DashboardController {

    @Autowired
    private DozerBeanMapper mapper;

    @Autowired
    private SubjectRepo subjectRepo;

    /**
     * Get all subjects when calling the given url of the rest api
     * @return a list of subjects
     */
    @RequestMapping(value = "/webresources/subject", method = RequestMethod.GET)
    public @ResponseBody
    List<Subject> getAllSubjects(){

        List<Subject> subjects = subjectRepo.findAll();

        return subjects;
    }
}
```

GradeController.java

```
package ch.post.pf.gradus.Controller;

import ch.post.pf.gradus.Models.Grade;
import ch.post.pf.gradus.Repositorys.GradeRepo;
import ch.post.pf.gradus.Response.Response;
import ch.post.pf.gradus.ViewModel.User.UserView;
import org.dozer.DozerBeanMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

/**
 * This class provides the methodes for the frontend to load everything that is needed for the grades.
 * @author manuel.bieri
 * @version 1.0.0 - 02.01.2017
 */
@RestController
@CrossOrigin
public class GradeController {

    @Autowired
    private DozerBeanMapper mapper;

    @Autowired
    private GradeRepo gradeRepo;

    /**
     * Create a new grade
     * @param userView the current view of the user
     * @return if the creation worked or not
     */
    @RequestMapping(value = "webresources/grade/create", method = RequestMethod.POST)
    public ResponseEntity<> createGrade(@RequestBody UserView userView) {

        Response createResponse = new Response();

        return new ResponseEntity<Response>(createResponse, HttpStatus.OK);
    }
}
```

SemesterController.java

```
package ch.post.pf.gradus.Controller;

import ch.post.pf.gradus.Models.Semester;
import ch.post.pf.gradus.Models.User;
import ch.post.pf.gradus.Repository.SemesterRepo;
import ch.post.pf.gradus.Repository.UserRepo;
import ch.post.pf.gradus.Response.Response;
import ch.post.pf.gradus.ViewModel.SemesterView;
import ch.post.pf.gradus.ViewModel.User.UserView;
import org.dozer.DozerBeanMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Collections;
import java.util.List;

/**
 * This class provides the methodes for the frontend to load everything that is needed for the semesters.
 * @author manuel.bieri
 * @version 1.0.0 - 03.01.2017
 */
@RestController
@CrossOrigin
public class SemesterController {

    @Autowired
    private DozerBeanMapper mapper;

    @Autowired
    private SemesterRepo semesterRepo;

    @Autowired
    private UserRepo userRepo;
```

```
/**
 * Create a new semester
 * @param semesterView the current semester view
 * @return if the creation worked or not
 */
@RequestMapping(value = "webresources/semester/create", method = RequestMethod.POST)
public ResponseEntity<?> createSemester(@RequestBody SemesterView semesterView) {

    Response createResponse = new Response();

    Semester semester = mapper.map(semesterView, Semester.class);

    semesterRepo.save(semester);

    return new ResponseEntity<Response>(createResponse, HttpStatus.OK);

}

/**
 * Get all semesters
 * @param userID the id of the user which is logged in
 * @return a list of all semesters and response if the query worked or not
 */
@RequestMapping(value = "/webresources/semester/{userID}", method = RequestMethod.GET)
public ResponseEntity<?> all(@PathVariable Long userID) {

    User creator = userRepo.findOne(userID);
    List<Semester> semesters = semesterRepo.findAllByCreator(creator);
    if (semesters.size() > 0) {
        return new ResponseEntity<List<Semester>>(semesters, HttpStatus.OK);
    } else {
        return new ResponseEntity<List<Semester>>(Collections.emptyList(), HttpStatus.NOT_FOUND);
    }

}

/**
 * Get one semester
 * @param id the id of the semester
 * @return the requested semester and response if the query worked or not
 */
@RequestMapping(value = "webresources/semester/one/{id}", method = RequestMethod.GET)
public ResponseEntity<?> one(@PathVariable String id) {
    if(id!="" && id.matches("\\d*"))
    {
        Long longId = Long.valueOf(id);
        Semester semester = semesterRepo.findOne(longId);
        if(semester!=null) {
            return new ResponseEntity<Semester>(semester, HttpStatus.OK);
        }
    }
    return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
}

}
```

SubjectController.java

```
package ch.post.pf.gradus.Controller;

import ch.post.pf.gradus.Models.Subject;
import ch.post.pf.gradus.Models.User;
import ch.post.pf.gradus.Repositorys.SubjectRepo;
import ch.post.pf.gradus.Repositorys.UserRepo;
import ch.post.pf.gradus.Response.Response;
import ch.post.pf.gradus.ViewModel.SubjectView;
import ch.post.pf.gradus.ViewModel.User.UserView;
import org.dozer.DozerBeanMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Collections;
import java.util.List;

/**
 * This class provides the methodes for the frontend to load everything that is needed for the subjects.
 * @author manuel.bieri
 * @version 1.0.0 - 03.01.2017
 */
@RestController
@CrossOrigin
public class SubjectController {

    @Autowired
    private DozerBeanMapper mapper;

    @Autowired
    private SubjectRepo subjectRepo;

    @Autowired
    private UserRepo userRepo;
```

```
/**
 * Create a new subject
 * @param subjectView the current semester view
 * @return the subject and if the creation worked or not
 */
@RequestMapping(value = "webresources/subject/create", method = RequestMethod.POST)
public ResponseEntity<?> createSubject(@RequestBody SubjectView subjectView) {

    Response createResponse = new Response();

    Subject subject = mapper.map(subjectView, Subject.class);

    subjectRepo.save(subject);

    return new ResponseEntity<Response>(createResponse, HttpStatus.OK);

}

/**
 * Get all subjects
 * @param userID the user id of the logged in user
 * @return the requested subjects and response if the query worked or not
 */
@RequestMapping(value = "/webresources/subject/{userID}", method = RequestMethod.GET)
public ResponseEntity<?> all(@PathVariable Long userID) {

    User creator = userRepo.findOne(userID);
    List<Subject> subjects = subjectRepo.findAllByCreator(creator);
    if (subjects.size() > 0) {
        return new ResponseEntity<List<Subject>>(subjects, HttpStatus.OK);
    } else {
        return new ResponseEntity<List<Subject>>(Collections.emptyList(), HttpStatus.OK);
    }

}

/**
 * Get one subject
 * @param id the id of the subject
 * @return the requested subject and response if the query worked or not
 */
@RequestMapping(value = "webresources/subject/one/{id}", method = RequestMethod.GET)
public ResponseEntity<?> one(@PathVariable String id) {
    if(id!="" && id.matches("\\d*"))
    {
        Long longId = Long.valueOf(id);
        Subject subject = subjectRepo.findOne(longId);
        if(subject!=null) {
            return new ResponseEntity<Subject>(subject, HttpStatus.OK);
        }
    }
    return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
}

}
```


UserController.java

```
package ch.post.pf.gradus.Controller;

import ch.post.pf.gradus.Models.User;
import ch.post.pf.gradus.Repositorys.UserRepo;
import ch.post.pf.gradus.Response.Response;
import ch.post.pf.gradus.ViewModel.User.UserView;
import org.dozer.DozerBeanMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.security.Principal;
import java.util.List;

/**
 * This class provides the methodes for the frontend to load everything that is needed for the user.
 * @author manuel.bieri
 * @version 1.0.0 - 06.01.2017
 */
@RestController
@CrossOrigin
public class UserController {

    @Autowired
    private DozerBeanMapper mapper;

    @Autowired
    private UserRepo userRepo;
```

```
/**
 * Create a new user
 * @param userView the current user View
 * @return the user and if the creation worked or not
 */
@RequestMapping(value = "webresources/user/create", method = RequestMethod.POST)
public ResponseEntity<?> createUser(@RequestBody UserView userView) {

    Response registrationResponse = new Response();

    registrationResponse.checkNotNull(userView.getFirstname(), "firstname is empty");
    registrationResponse.checkNotNull(userView.getLastname(), "lastname is empty");

    registrationResponse.checkNotNull(userView.getEmail(), "email is empty");
    registrationResponse.checkIfNotEmail(userView.getEmail(), "email in a not valid format");

    registrationResponse.checkNotNull(userView.getPassword(), "password is empty");

    registrationResponse.checkIfNotStrongPW(userView.getPassword(), "password to weak");

    if(!registrationResponse.getState()) {

        User sameEmailUser = userRepo.findOneByEmail(userView.getEmail());
        registrationResponse.checkIfObjectNotNull(sameEmailUser, "email already used");

        if(!registrationResponse.getState()) {

            User user = mapper.map(userView, User.class);
            userRepo.save(user);

            registrationResponse.setMessage(user.getId().toString());

        }
    }

    return new ResponseEntity<Response>(registrationResponse, HttpStatus.OK);
}

/**
 * Get all users
 * @return a user list and response if the query worked or not
 */
@RequestMapping(value = "/webresources/user", method = RequestMethod.GET)
public @ResponseBody
List<User> getAllUser(){

    List<User> users = userRepo.findAll();

    return users;
}
```

```
/**
 * Get one user
 * @param id the id of the user
 * @return the requested user and response if the query worked or not
 */
@RequestMapping(value = "webresources/user/{id}", method = RequestMethod.GET)
public ResponseEntity<?> one(@PathVariable String id) {
    if(id!="" && id.matches("\\d*"))
    {
        Long longId = Long.valueOf(id);
        User user = userRepo.findOne(longId);
        if(user!=null) {
            return new ResponseEntity<User>(user, HttpStatus.OK);
        }
    }
    return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
}

/**
 * Sign in the user.
 * @param userView the current user view
 * @return response if the user new is logged in or not
 */
@RequestMapping(value = "webresources/user/signin", method = RequestMethod.POST)
public ResponseEntity<?> signIn(@RequestBody UserView userView) {

    Response response = new Response();

    User user = userRepo.findOneByEmail(userView.getEmail());

    response.checkIfObjectNull(user, "Email / Password not correct");

    if (!response.getState()) {

        response.checkIfNotEqual(user.getPassword(), userView.getPassword(), "Email / Password not correct");

        if (!response.getState()) {

            response.setMessage(user.getId().toString());

        }
    }

    return new ResponseEntity<Response>(response, HttpStatus.OK);
}
}
```

GradeRepo.java

```
package ch.post.pf.gradus.Repositorys;

import ch.post.pf.gradus.Models.Grade;
import ch.post.pf.gradus.Models.Semester;
import ch.post.pf.gradus.Models.Subject;
import ch.post.pf.gradus.Models.User;
import org.springframework.data.repository.CrudRepository;

import java.util.List;

/**
 * This class provides the methodes to get the needed data of the grades
 * @author manuel.bieri
 * @version 1.0.0 - 02.01.2017
 */
public interface GradeRepo extends CrudRepository<Grade, Long> {

    /**
     * Find all grades
     * @return list of grades
     */
    List<Grade> findAll();

    /**
     * Find all grades by the creator
     * @param creator the creator of the grade
     * @return list of grades
     */
    List<Grade> findAllByCreator(User creator);

    /**
     * Find all grades by the creator and a semester
     * @param creator the creator of the grade
     * @param semester the semester
     * @return list of grades
     */
    List<Grade> findAllByCreatorAndSemester(User creator, Semester semester);

    /**
     * Find all grades by the creator and a subject
     * @param creator the creator of the grade
     * @param subject the subject
     * @return list of grades
     */
    List<Grade> findAllByCreatorAndSubject(User creator, Subject subject);

}
```

SemesterRepo.java

```
package ch.post.pf.gradus.Repositories;

import ch.post.pf.gradus.Models.Semester;
import ch.post.pf.gradus.Models.User;
import org.springframework.data.repository.CrudRepository;

import java.util.List;

/**
 * This class provides the methodes to get the needed data of the semesters
 * @author manuel.bieri
 * @version 1.0.0 - 02.01.2017
 */
public interface SemesterRepo extends CrudRepository<Semester, Long> {

    /**
     * Find all semesters
     * @return list of semesters
     */
    List<Semester> findAll();

    /**
     * Find all semesters by the creator
     * @param creator the creator of the semester
     * @return list of semesters
     */
    List<Semester> findAllByCreator(User creator);

}
```

SubjectRepo.java

```
package ch.post.pf.gradus.Repositorys;

import ch.post.pf.gradus.Models.Subject;
import ch.post.pf.gradus.Models.User;
import org.springframework.data.repository.CrudRepository;

import java.util.List;

/**
 * This class provides the methodes to get the needed data of the subjects
 * @author manuel.bieri
 * @version 1.0.0 - 02.01.2017
 */
public interface SubjectRepo extends CrudRepository<Subject, Long> {

    /**
     * Find all subjects
     * @return list of subjects
     */
    List<Subject> findAll();

    /**
     * Find all subjects by the creator
     * @param creator the creator of the subject
     * @return list of subjects
     */
    List<Subject> findAllByCreator(User creator);
}
```

UserRepo.java

```
package ch.post.pf.gradus.Repositorys;

import ch.post.pf.gradus.Models.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;

import java.util.List;
import java.util.Set;

/**
 * This class provides the methodes to get the needed data of the user
 * @author manuel.bieri
 * @version 1.0.0 - 02.01.2017
 */
public interface UserRepo extends CrudRepository<User, Long> {

    /**
     * Find all users
     * @return list of users
     */
    List<User> findAll();

    /**
     * Find a user by his lastname and his firstname
     * @return the searched user
     */
    User findByLastnameAndFirstname(String lastname, String firstname);

    /**
     * Find a user by his email
     * @return the searched user
     */
    User findOneByEmail(String email);

}
```