

Enroll.me

arc42 Dokumentation

Arotăriței Vlad Badea Patrick
Gati Mark Petrean Simona Titieni Alexandru

Inhalt

1. Einleitung und Ziele	4
1.1. Bedingungen Überblick.....	4
1.2. Qualitätsziele	5
1.3. Interessenvertretern	5
2. Randbedingungen	7
2.1. Technischen Randbedingungen.....	7
2.2. Organisatorische Randbedingungen.....	7
2.3. Konventionen	8
3. Kontextabgrenzung	9
3.1. Fachlicher Kontext.....	9
3.2. Verteilungskontext	10
4. Lösungsstrategie.....	12
5. Bausteinsicht	13
5.1. Niveau 1.....	13
5.2. Teilsystem View (graphische Schnittstelle)	14
5.3. Teilsystem Controller.....	15
5.4. Teilsystem Repository.....	16
5.5. Teilsystem Model.....	18
6. Laufzeitsicht.....	19
7. Verteilungsschicht	20
8. Konzepte.....	21
8.1. Abhängigkeiten.....	21
8.2. Domain Model.....	21
8.3. Benutzerschnittstelle.....	22
8.4. Validierung	24
8.5. Fehlerbehandlung	24
8.6. Testbarkeit.....	25

9. Entwurfsentscheidungen.....	26
10. Qualitätszenarien	27
10.1. Dienstprogrammbaum	27
10.2. Bewertungsszenarien	28
11. Risiken	29
11.1. SQL-Injection	29
11.2. Aufgeteilte Controller und schlimme Abtrennung der UI von der Logik.....	30
12. Glossar.....	31
12.1. Einleitung.....	31
12.2. Ausdrücke und Begriffe.....	31

KAPITEL 1

Einleitung und Ziele

1.1. Bedingungen Überblick

Was ist Enroll.me?

- Enroll.me ist ein Programm, mit dem man eine Datenbank von Studenten, Professoren und Vorlesungen basierend auf verschiedenen Kriterien zugreifen kann.
- Es hat eine organische, leichtbenutzbare grafische Schnittstelle.
- Es kann sowohl für Studenten, als auch für Professoren benutzt werden ohne Veränderungen in der Arbeits-Flow.

Essentiellen Eigenschaften

- Eine grafische Schnittstelle für die Studenten indem man in Vorlesungen anmelden kann und ECTS sehen
- Eine grafische Schnittstelle für die Professoren indem man alle Studenten die in die Vorlesungen, die der Professor unterrichtet, angemeldet sind sehen
- Einen Login-Bildschirmfenster für einloggen
- SQL-Datenbank-Unterstützung
- Leichtbenutzbar und ähnlich anderen grafischen Schnittstellen, die denselben Zweck dient

1.2. Qualitätsziele

Die folgende Tabelle beschreibt die Qualitätsziele für Enroll.me, in der Ordnung der Priorität.

Qualitätsziel	Beschreibung
Leicht benutzbar	Der Programm soll im Allgemein keinen Lernkrue brauchen. Die Anwendung soll mit andere Programme die das gleiche Ziel haben meist ähnlich sein.
Bug-frei	Seiend eine einfache, aber auch eine wichtige Teil von Uni-Management, Enroll.me soll, so viel wie möglich, Bug-frei sein.
Leicht verwaltbar	Der Programm soll, so viel wie möglich, keinen Verwaltung oder Verweiterung brauchen.
Bestimmte Zielgruppe	Die Anwendung hat eine doppelte Zielgruppe: es is sowohl zu Professoren, als auch zu Studenten gerichtet.
Schnelle Rückantwort	Die Anwendung soll geschwindigkeitsweise performant sein.

1.3. Interessenvertretern

Die folgende Tabelle zählt alle Interessenvertretern und ihre Interesse ein.

Wer?	Beschreibung
Entwicklers	<ul style="list-style-type: none"> • Gruppenarbeit zu verstärken • Java lernen • graphischen Schittstellen darstellen zu lernen • eine gute Note kreigen
Universität „Babeş-Bolyai“	<ul style="list-style-type: none"> • bietet uns Support und Materialien

Holger Klus	<ul style="list-style-type: none">• will eine vollständige arc42 Dokumentation sehen• bietet uns Support und Materialien
-------------	---

KAPITEL 2

Randbedingungen

2.1. Technischen Randbedingungen

Beschränkung	Motivation
Entwickelt und getestet auf Windows	Keinen Teammitglied benutzt einen Betriebssystem außer Microsoft Windows, aber der Software muss Betriebssystemfrei sein.
Implementiert in Java	Java bietet eine Betriebssystemfreie Plattform für Software-Entwicklung.
Minimale Abhängigkeiten	Enroll.me soll in besten Fall nur JavaFX und JDBC als externen Libraries benutzen. Wenn externen Libraries benutzt werden, sollen diese mindestens kostenlos und open-source sein.
Open-Source	Die Anwendung wird kostenlos und open-source auf GitHub hochgeladen sein.

2.2 Organisatorische Randbedingungen

Beschränkung	Motivation
COVID-19 Pandemie	Keinen Treffen außer Online-Meetings möglich. Treffen werden Online auf Skype oder auf einen anderen Plattform stattfinden. Alle Prioritätsnachrichten werden durch WhatsApp geschickt.
Zeitplan und Termin	Die Dokumentierung hat 22 Mai begonnen. Der Abschluß den Projekt muss 26 Mai 2020 am 3 PM stattfinden.
Verteilung der Aufgaben	Die Teilen dieser Dokumentation wird zwitschen den Mitglieder verteilt.

Testen	JUnit wird für den Entwicklung der Testen benutzt.
Abschluß und Veröffentlichung der Kode	Der Kode soll auf GitHub öffentlich eingeladen sein. Der GitHub-Projekt soll auch externe Scripts (wie z.B. der SQL-Anfrage für das Datenbankkonstruierung) die während der Entwicklungsphase benutzt waren enthalten. Die arc42 wird auch in den einen separaten Folder (z.B. docs) hochgeladen.

2.3 Konventionen

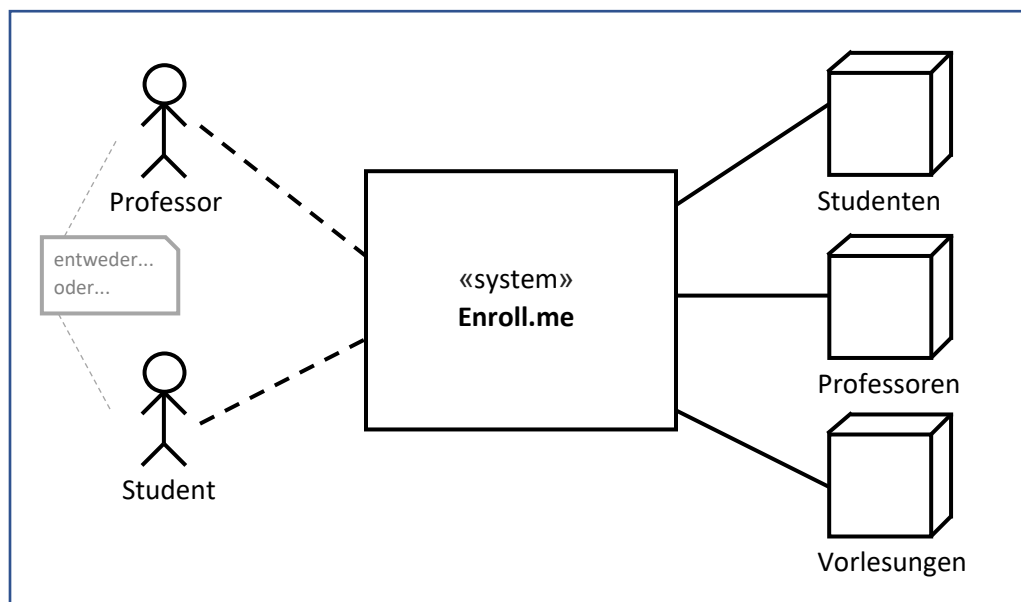
Konvention	Motivation
Dokumentation der Arhitektur	Die Dokumentation der Arhitektur muss auf den arc42 Version 7.0 Muster beachten.
Name der Professoren, Vorlesungen und Studenten	Die Namen der Professoren, Vorlesungen und Studenten müssen, so viel wie möglich, reelen Daten von Universität Babeş-Bolyai verspiegeln.
Kodeart	Die Kode wird das Google Java Style Guide beachten.

KAPITEL 3

Kontextabgrenzung

3.1. Fachlicher-Kontext

Die folgende Schema erklärt die Kommunikation zwischen die Softbestandteile der Anwendung auf einem PC mit Windows Betriebssystem.

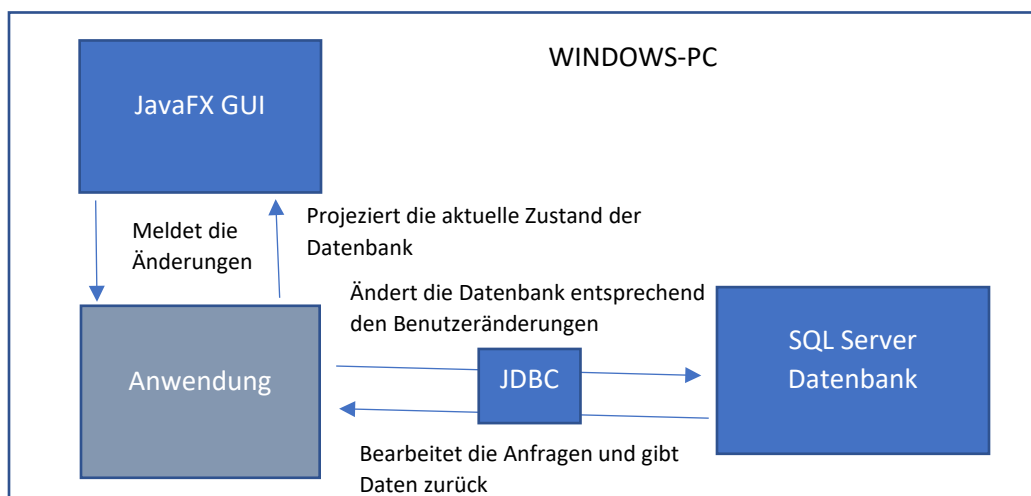


Nachbar	Beschreibung
Professoren (Benutzer)	Der Professor fügt oder löscht einen Vorlesung von den Datenbank ein, verändert der ECTS Nummer und schaut die Studenten, die an einer der von den Professor unterrichteten Vorlesungen eingeschrieben sind, an.

Student (Benutzer)	Der Student kann sowohl seinen ECTS und Vorlesungen anschauen, als auch sich an neue Vorlesungen einschreiben.
Vorlesungen	Teoretische Einheit die Beziehungen sowohl auf Professoren als auch auf Studenten hat. Eine Vorlesung ist von einen Professor unterrichtet und hat mehrere eingeschriebene Studenten.
Anmeldung	Der Fall indem der Student eine Beziehung zu einen Vorlesung konstruiert.

3.2. Verteilungskontext

Die folgende Schema erklärt die Kommunikation zwischen die Softbestandteile der Anwendung auf einem PC mit Windows Betriebssystem.



Nachbar	Beschreibung
Datenbank	Externes SQL-Datenbank
JavaFX	Graphisches Schnittstelle für Java-Anwedungen. Die Entwicklung von JavaFX ist nicht einen Teil von Enroll.me.

JDBC	Enroll.me erlaubt einen optionellen zugriff zu SQL-Datenbanken mit JDBC. Die Entwicklung von JDBC ist nicht einen Teil von Enroll.me.
------	---

KAPITEL 4

Lösungsstrategie

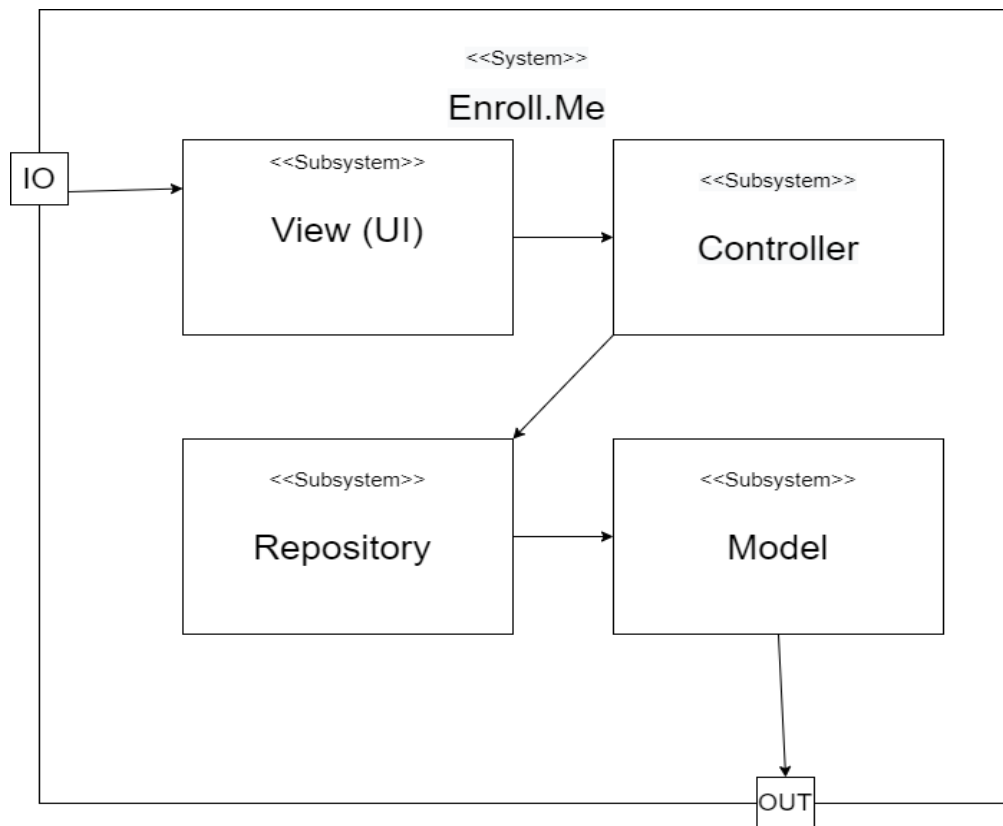
Qualitätsziel	Dem zuträgliche Ansätze in der Architektur
Leichte Verwendung der Anwendung	<ul style="list-style-type: none">• Java als Programmiersprache, die weiterhin die Umsetzung von JavaFX als GUI erlaubt.
Schnelle Rückantwort	<ul style="list-style-type: none">• Eine lokale SQL Server Datenbank ermöglicht eine latenzfreie Verbindung mit der Anwendung• JDBC schafft die Kommunikation zwischen der Datenbank und dem Klient.
An Benutzerkategorie gerichtet	<ul style="list-style-type: none">• Zwei getrennte View-Implementierungen, sowohl für die Lehrkräfte, als auch für die Studenten, wo man FXML benutzt wurde.
Bugfrei	<ul style="list-style-type: none">• Minimale Softwareabhängigkeiten und intensive manuelle Überprüfung der Anwendung
Einfache Verwaltung	<ul style="list-style-type: none">• Durch vollständige und klare Architektur

Eine Schichtenarchitektur wurde umgesetzt, um eine geringere Kopplung zwischen die Modulen zu erreichen, beziehungsweise eine hohe Kohäsion zu schaffen. Diese Architektur erlaubt uns eine einfache Refaktorisierung in dem Fall man muss Änderungen oder Verbesserungen in der Anwendung bringen.

KAPITEL 5

Bausteinsicht

5.1. Niveau 1

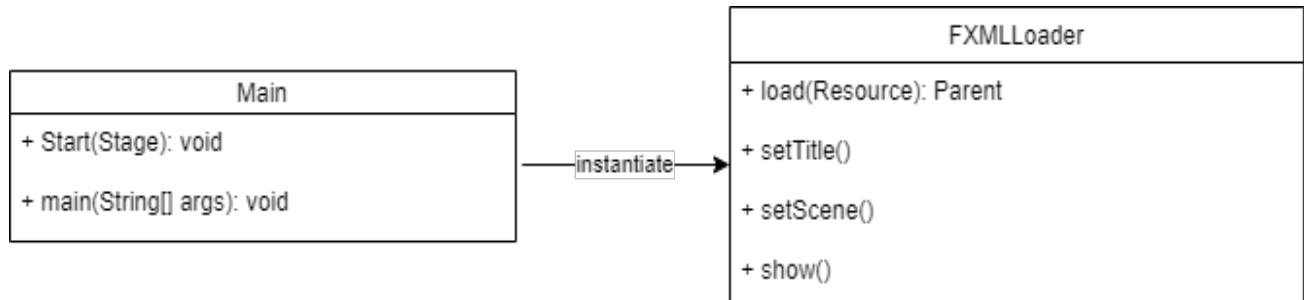


Enroll.Me gliedert sich in vier Subsysteme, wie unten dargestellt. Die gestrichelten Pfeile stellen logische Abhängigkeiten zwischen den Subsystemen dar ("x -> y" für "x hängt von y ab"). Die quadratischen Kästchen auf der Membran des Systems sind Interaktionspunkte („Ports“) mit der Außenwelt

Subsystem	Beschreibung
View (UI)	Realisiert die Kommunikation mit einem Client mithilfe des Enroll.Me-Protokolls

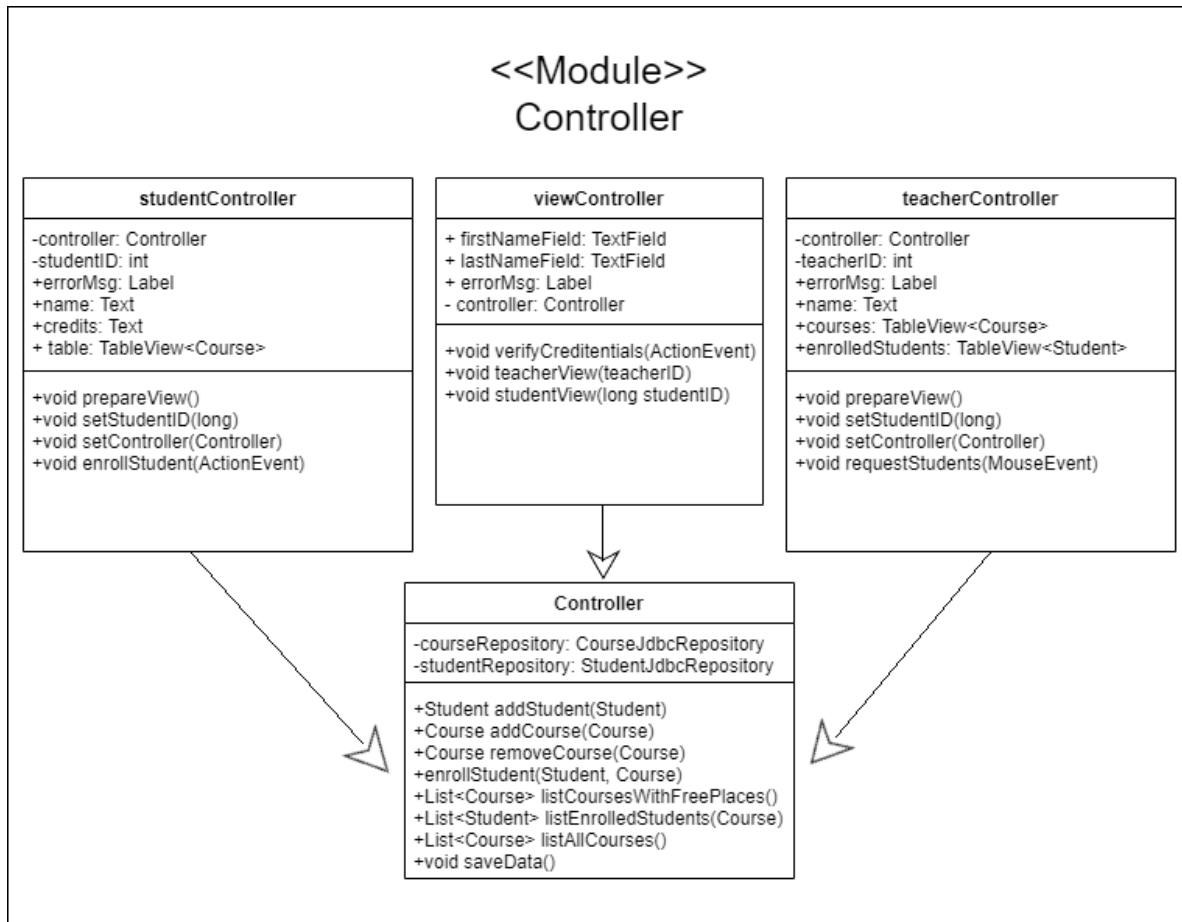
Controller	Bietet Funktionen für den Client
Repository	Stellt eine Verbindung zur Datenbank her und kann Änderungen vornehmen
Model	Beschreibung der gespeicherten Objekte

5.2 Subsystem View (UI) (Blackbox)



Dieses Subsystem implementiert die Kommunikation mit einem Client (Schüler oder Lehrer), der von einem FXML-Fenster mit dem Namen loginWindow festgelegt wurde. Basierend auf den angegebenen Anmeldeinformationen wird entweder eine grafische Benutzeroberfläche für Lehrer oder Schüler geöffnet, indem eine Verbindung mit dem Controller verwendet wird.

5.3 Subsystem Controller (Blackbox)

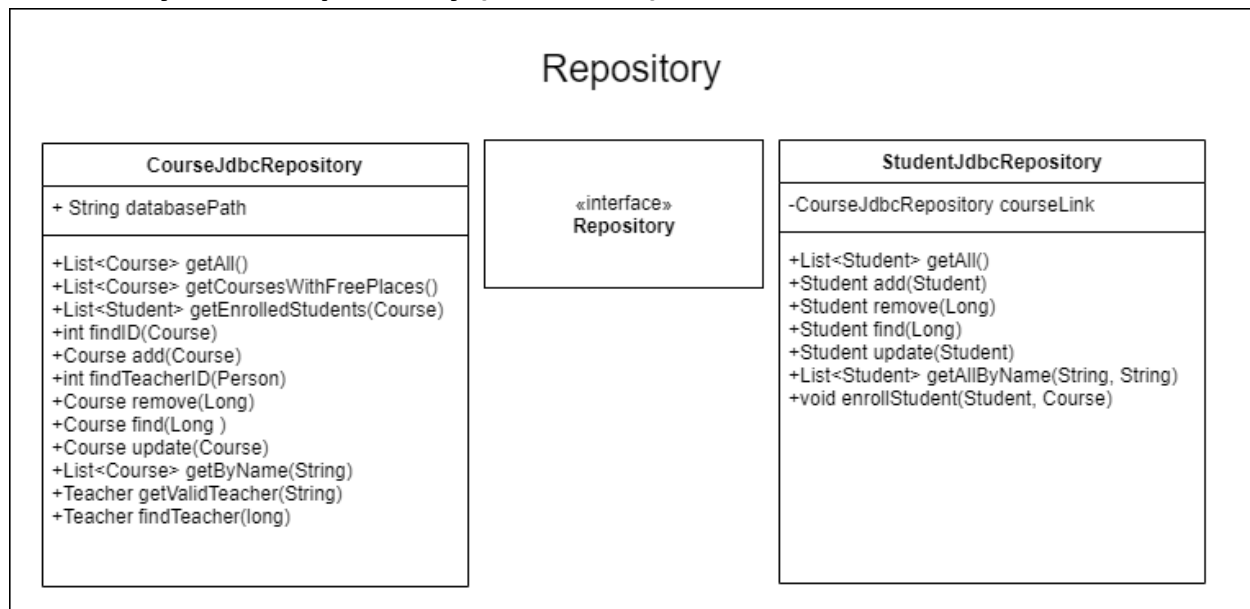


Die Controller ist die Subsystem die alle anderungen in den Datenbank und funktionalitaeten der Student/Lehrer organisiert und verifiziert, falls etwas nicht passt dann ist es ab zu den Controller es zu entscheiden was man tut.

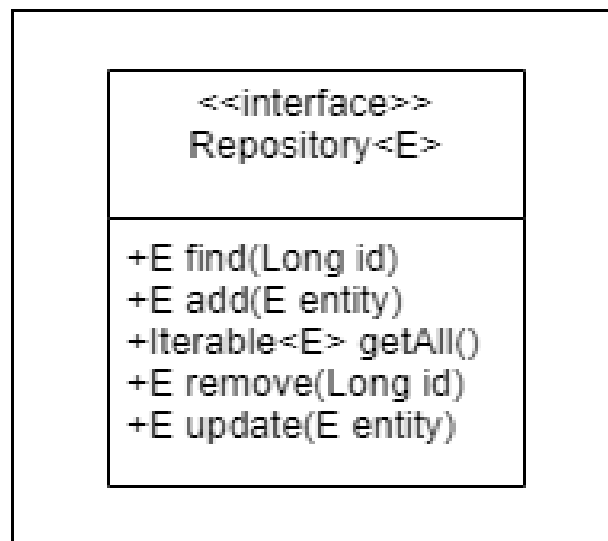
Methoden	Beschreibung
PrepareView()	Einfügt alle Vorlesungen/Studenten zu der Tabelle
setStudentID()	Einstellt die StudentenID von der loginWindow
setController()	Einstellt die primäre Controller zu der Klasse
enrollStudent()	Description of the stored objects

verifyCredentials()	Überprüft ob die eingegebene Referenzen im datenbank sind
teacherView()	Einstellt die Protokolle wie die Daten gesehen werden von der Lehrer
studentView()	Einstellt die Protokolle wie die Daten gesehen werden von der Student
setTeacherID()	Einstellt die LehrerID von der loginWindow
requestStudent()	Einfugt alle Studenten die zu den Vorlesungen der Lehrer angemeldet sind zu der Tabelle
addStudent()	Einfugt ein neues Student
addCourse()	Einfugt ein neues Vorlesung
removeCourse()	Loscht ein Vorlesung
saveData()	Speichert die Repository in dem Datenbank

5.4 Subsystem Repository (Blackbox)

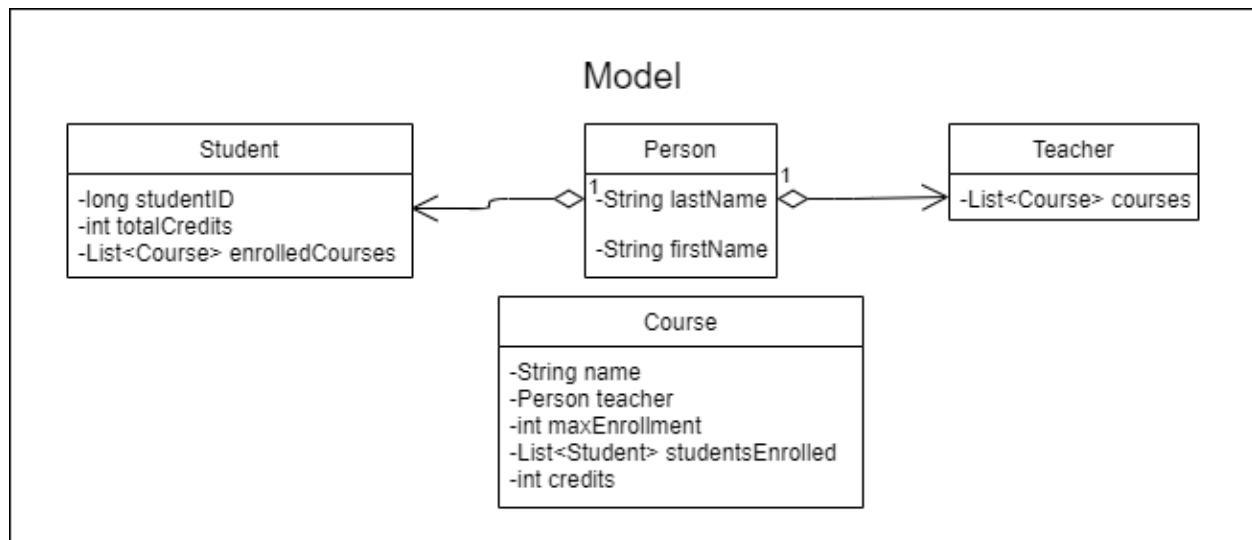


Methoden	Beschreibung
getAll()	Gibt alle Studenten/Lehrer in der Datenbank zurück
getCoursesWFreePla()	Gibt alle Vorlesungen mit freien Plätze
getEnrolledStudents()	Gibt alle Studenten, die zu einen bestimmten Vorlesung angemeldet sind
add()	Fügt einen Student/Lehrer in der Datenbank ein
remove()	Loscht einen Student/Lehrer
find()	Sucht einen Student/Lehrer in der Datenbank durch ID
update()	Aktualisiert einen Student/Lehrer
getValidTeacher()	Gibt ob einen Lehrer is valid (durch name) zurück
enrollStudent()	Fügt ein neues Paar von Vorlesung-Student in der Datenbank ein



Die Student/Course JDBCRepository warden eine gemeinsame interface implementiere, `<<Repository>>`, die funktionalitat für CRUD Operationen bietet.

5.5 Subsystem Model (Blackbox)



KAPITEL 6

Laufzeitsicht

Nach dem Einrichten des LoginView-Protokolls startet der Client die Authentifizierung, indem er den Vor- und Nachname eines Students oder Lehrers angibt.

Das folgende Sequenzdiagramm zeigt eine beispielhafte Interaktion auf Subsystemebene von der Eingabe „Popescu Andreea“ bis zur Anzeige der Kurse, an denen der Student teilnimmt.

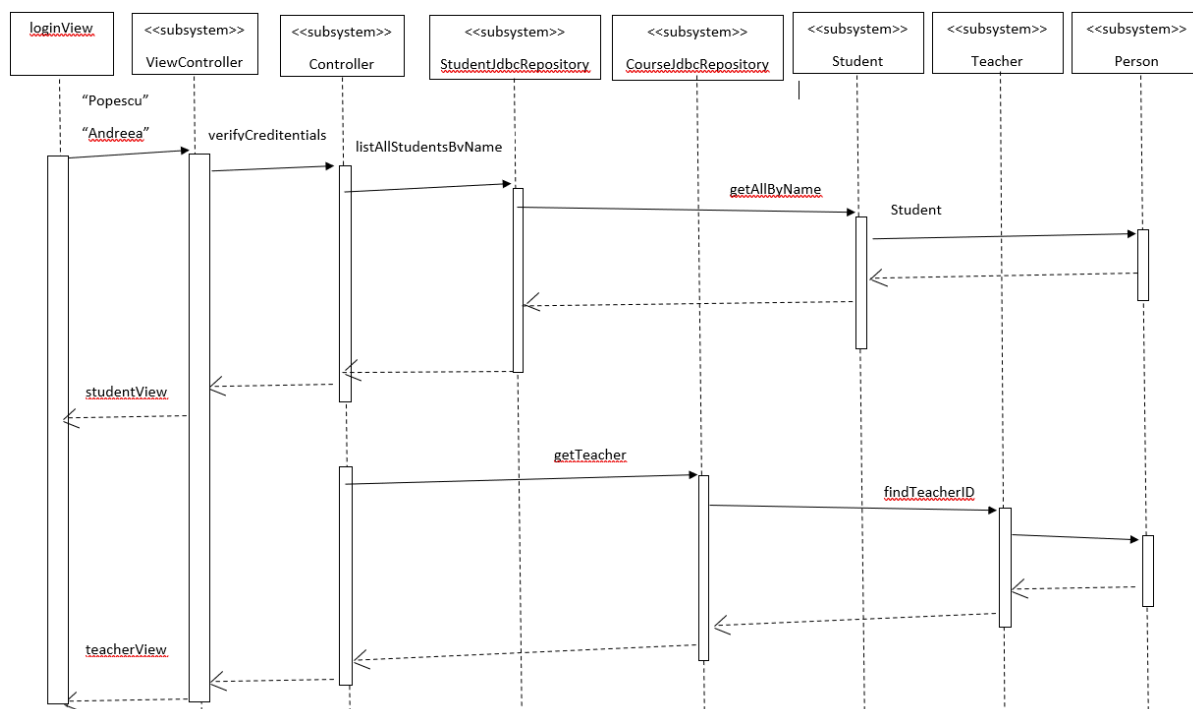
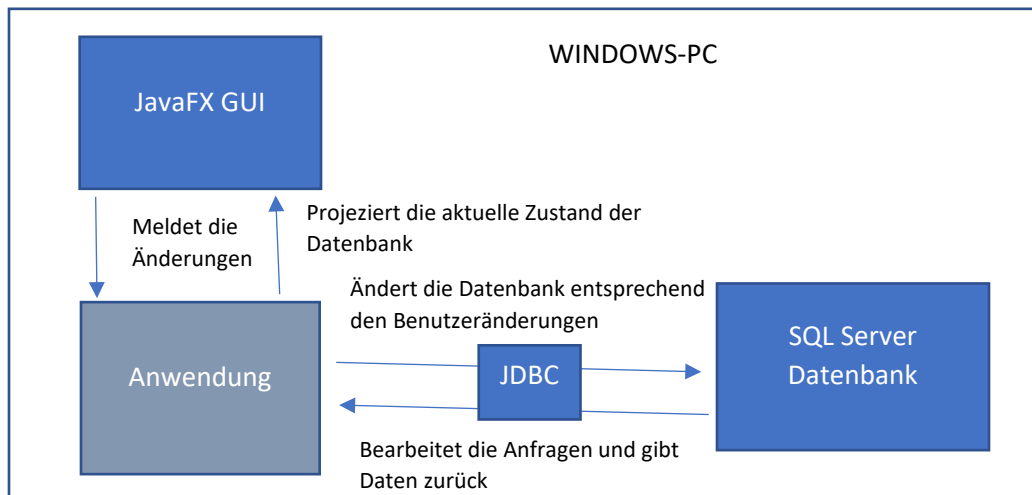


Abb.: Beispielinteraktion zur Bewegungsbestimmung

Zunächst überprüft das *ViewController*-Subsystem, ob der Student mithilfe des *Controller*-Subsystems existiert. Wenn der Student existiert, werden die Lehrer und die Kurse angezeigt, in denen er eingeschrieben ist.

KAPITEL 7

Verteilungsschicht



Softwareanforderungen auf PC:

- Java Runtime Environment SE 7 (oder höher)
- SQL Server 2016 (oder höher)
- Microsoft JDBC Driver

Um eine erfolgreiche Ausführung der Anwendung zu schaffen, muss man im Code das Verbindungsstring mit dem eigenen Name des Servers ändern und die entsprechenden Tabelle darin erstellen.

KAPITEL 8

Konzepte

8.1 Abhängigkeiten

Die Abhängigkeiten zwischen den Modulen sind quasi primitiv, in der Sinne, dass obwohl die Anwendung als eine Schichtenarchitektur gestaltet ist, es gibt Referenzen in der high level Modulen zu Objekten der Modulen die 2 Niveaus niedriger sind.

Bei dem Repository Niveau haben alle tatsächlichen Repositories dieselbe Schnittstelle implementiert, nicht um z.B. Wiederverwendbarkeit des Codes sicherzustellen, sondern dasselbe Verhalten umzusetzen.

Die Abhängigkeit zwischen der Datenbank und der Java Anwendung wird durch ein JDBC Driver erfüllt.

8.2 Domain Model

Die niedrigste Schicht, das Modell, enthält alle Objekte daraus Entitäten schaffen wird - Teacher und Student vererben von Person, beziehungsweise Course.

Die nächste Schicht ist das Repository - da steht ein StudentRepository, dass die Studenten verwaltet, beziehungsweise ein CourseRepository, dass sowohl Courses als auch Teachers gemeinsam organisiert, weil Teachers und Courses existenzabhängig sind.

Das Controller steht weiterhin höher in der Hierarchie und besteht aus:

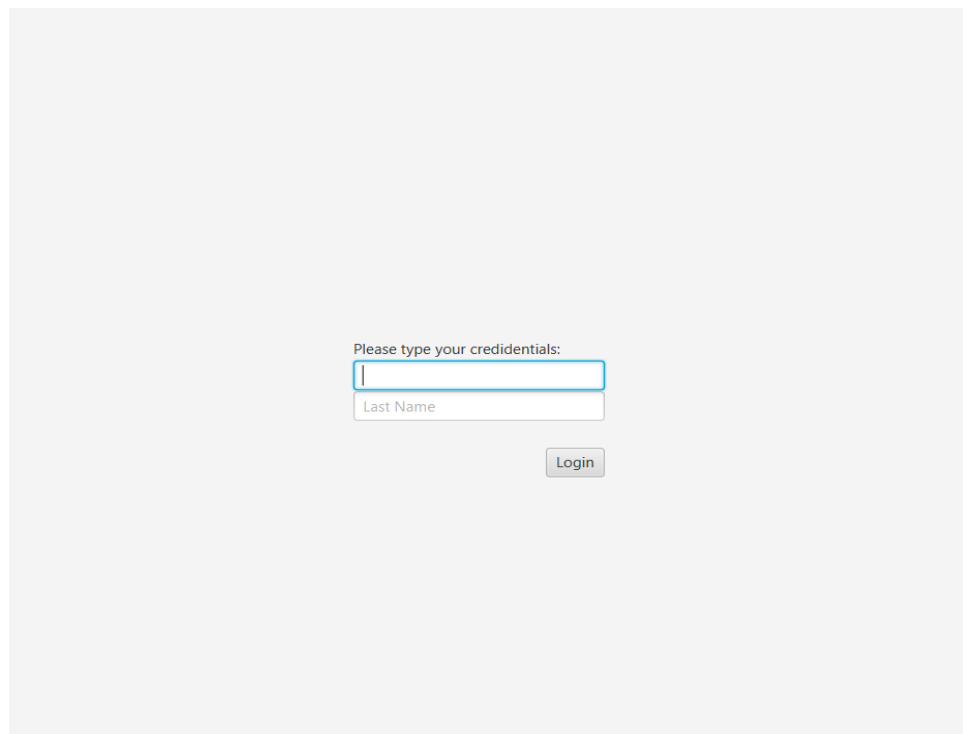
- Die Klasse Controller – behandelt die Systemereignisse, in dem man entsprechend die Zustand der Repositories ändert.
- Student- und Teachercontroller – zuständig für die Vorbereitung der entsprechenden Views.

Die letzte Schicht ist das View und besteht aus : ViewController – enthält Methoden für Loginüberprüfung und Erstellung der GUI Fenster für Students/Teachers.

8.3 Benutzerschnittstelle

Das GUI Bestandteil wird durch JavaFX Fenster veranschaulicht und es gibt insgesamt 3 solche Fenster:

Das erste Fenster besteht aus einem einfachen Login, die abhängig von ob es sich ein Student oder Teacher anmeldet, ein der folgenden Fenster weitererscheinen wird.



Please type your credentials:

Last Name

Login

Das Fenster für Studenten wird alle Courses anzeigen und der Student kann sich an je welchen anmelden, danach ändert sich auch die Anzahl der Kredite entsprechend.

Alex Titieni (1)

Number of credits: 22

Course	Teacher	Places	Credits
Datenbanken	Cristea Diana		5
Fortgeschrittene Programmierung	Iulian Manda		5
Objekt Orientierte Programmierung	Sebi Gmandt		6
Logische Programmierung	Cristea Diana		10
Künstliche Intelligenz	Iulian Manda		5
Web Programmierung	Sebi Gmandt		4
Datenstrukturen und Algorithmen	Cristea Diana		6
Betriebssysteme	Iulian Manda		5

< >

Enroll

Die Fenster für Teachers zeigt links alle Courses, die ein Teacher veranstaltet, und rechts alle Studenten die bei dem Course angemeldet sind.

[illegible]

Cristea Diana

[illegible]

8.4 Validierung

Es ist keine Validierung erforderlich, weil der Kunde keine neuen Personen in die Anwendung einführen kann.

8.5 Fehlerbehandlung

Alle Subsystemmethoden lösen SQL- Ausnahmen aus.

Wenn wir den Vor- und Nachname eines Students oder Lehrers eingeben, wird geprüft, ob er in der Datenbank vorhanden ist. Wenn es nicht existiert, wird eine Warnmeldung angezeigt.

Please type your credentials:

There is no person with this name

Login

Ein Meldung wird in der Konsole angezeigt, wenn:

- ein Student die Anzahl von 30 Kredite überschreitet
- für einen Kurs keine Plätze mehr verfügbar sind
- ich mich an einen Kurs teilnimmt anmelden möchte an dem ich schon angemeldet in

```
Der Vorlesung hat keine Platze frei.
Der Vorlesung hat keine Platze frei.
Der Zahl der Kredite wurde ubergeschritten.
Der Zahl der Kredite wurde ubergeschritten.
```

8.6 Testbarkeit

Die Funktionalität der einzelnen Module von Enrolled.me wird durch umfangreiche Unit-Tests sichergestellt. Sie finden einen Ordner src/test neben src/controller, in dem der Java-Quellcode der Module gespeichert ist. Es spiegelt die Paketstruktur wider und testet in den entsprechenden Paketen Unit-Tests für die mit realisierten Klassen Junit 5.

Standard-Unit-Tests, die die einzelnen Klassen untersuchen, warden als Klasse selbst mit dem Suffix Test bezeichnet.

```
public class ControllerTest {
    private CourseRepository courseRepository;
    private StudentRepository studentRepository;

    @Before
    public void CourseRepositoryTest() throws AlreadyInListException {
        this.courseRepository = new CourseRepository();
        this.courseRepository.add(new Course("MAP", new Teacher("Catalin", "Rusu"), 30, 15));
        this.courseRepository.add(new Course("BD", new Teacher("Diana", "Troanca"), 41, 11));
        this.courseRepository.add(new Course("Statistica", new Teacher("Hannelore", "Brekner"), 16, 7));

        this.studentRepository = new StudentRepository();
        this.studentRepository.add(new Student("John", "Doe", 220));
        this.studentRepository.add(new Student("Jane", "Doe", 221));
        this.studentRepository.add(new Student("James", "Hamilton", 222));
    }
}
```

KAPITEL 9

Entwurfsentscheidungen

Enroll.me soll eine SQL-Datenbank mit einer graphischen Schnittstelle, die Veränderungs-Operationen auf der Datenbank führen können, verkoppeln.

Argumente für die Entscheidung

Die Architektur ist auf eure vorherige Erfahrung mit Java und insbesondere JavaFX, die manchmal seine eigene Struktur-Muster braucht. Die Hinweise beziehungsweise Architektur von den vorherigen Schuljahren spielen auch eine wichtige Rolle hier. Die Anwendung muss leicht verwaltbar sein und das bietet eine klare und einfache Architektur. Die technischen Randbedingungen (siehe 2.1) werden auch streng beachtet.

Alternativen

Unter diesen Bedingungen, die Qualitätsziele können sowohl mit alternativer Datenbanksoftware wie MySQL erreicht werden, aber das bietet einen lokalen Server einzustellen und das wird der Prinzip der Einfachheit verletzen.

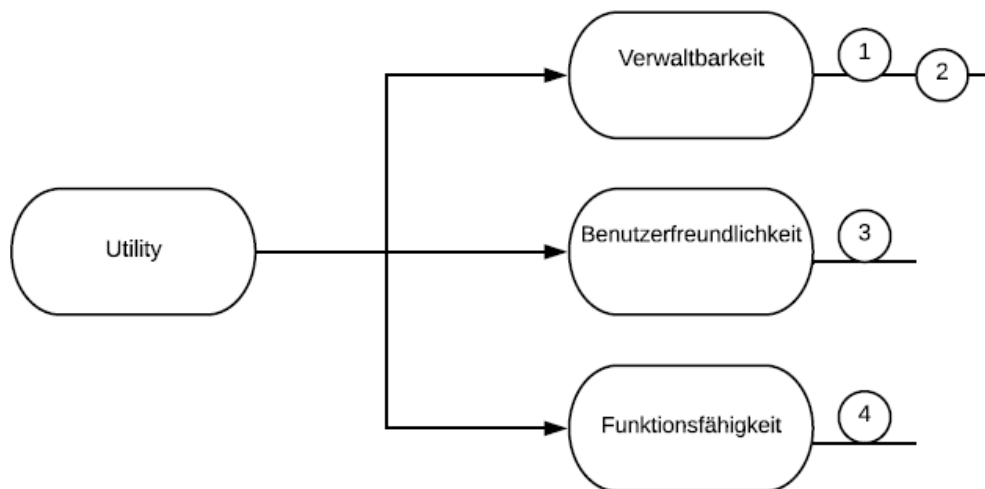
KAPITEL 10

Qualitätsszenarien

Die Qualitätsszenarien in diesem Abschnitt zeigen die grundlegenden Qualitätsziele sowie andere erforderliche Qualitätseigenschaften.

10.1. Utility Tree

Das folgende Diagramm gibt einen Überblick über die relevanten Qualitätsmerkmale und die damit verbundenen Szenarien.



10.2. Bewertungsszenarien

1. Ein Java Entwickler oder ein Datenbankarchitekt will sich die Datenbank des Projektes anschauen um zu überprüfen ob die Daten richtig gespeichert werden, also ob sich Studenten für Kursen anmelden können, nachdem er das bestätigt kann er die Anwendung

erweitern indem er z.B. neue Tabellen für andere Informationen erschafft.

2. Eine Universität braucht eine Anwendung um Studenten an Kurse zu registrieren, die aus finanziellen Gründen keine Angestellte benötigen soll, die die Anwendung regelmäßig überprüfen. Diese Anwendung bietet eine einfache Lösung dafür.
3. Ein Student will sich an bestimmte Kurse anmelden, er soll die Kurse sich einfach anschauen können. Dazu sollte er nicht Stunden lang mit der Anwendung experimentieren, sondern alles soll einfach und intuitiv sein damit er keine Fähler macht
4. Ein Student möchte sich z.B. für Lineare Algebra anmelden, er kann das einfach mit die Anwendung Enroll.Me machen.

KAPITEL 11

Risiken

Eine Universität Die folgenden Risiken wurden identifiziert, aber noch nicht repariert. Diese Architekturübersicht erklärt was diese Risiken sind und wie man sie vermeidet und zeigt auch eventuelle Korrekturen der Architektur oder Codes.

11.1. SQL Injection

In diese Iteration des Projektes, kann der Benutzer verschiedene Prozeduren, Indexen, Daten oder sogar ganze Tabellen aus dem Datenbank löschen, welche die Leistung stark beeinflussen könnte, oder die ganze Programmausführung gefährdet.

Eventualitätsplanung

Man könnte die ganze Datenbank in eine Liste von Benutzer am Anfang speichern, und beim Anmelden einfach zu überprüfen ob der Benutzer in der Liste ist anstatt eine SQL-Abfrage dynamisch aufzubauen.

Da diese Anwendung für einen kleineren Publikum gedacht ist könnte dies gut funktionieren, aber je mehrere Benutzer, desto mehr Speicherplatz wird benötigt und längeren Laufzeit.

Risiko Minimierung

Man kann SQL Injection vermeiden, indem man gespeicherte Prozeduren mit Parametern benutzt, die keine dynamisch erzeugte Abfragen beinhalten oder zum Beispiel für unser Fall:

Man könnte JPA's Query API benutzen, um eine JPA query method mit CriteriaBuilder zu erzeugen.

11.2. Aufgeteilte Kontroller und schlimme Abtrennung der UI von der Logik

Der MVC Designmuster (Model-view-controller design pattern) besagt, dass es nur ein einziger Kontroller sein soll, der sich mit der ganzen Logik des Programmes beschäftigt und somit die Benutzer Ebene, also den UI von den Repository trennt. In diese Iteration des Projektes, gibt es 4 Kontroller: ein Main Kontroller, ein UI Kontroller und je einen Kontroller für die Lehrer und einen für die Studenten.

Dies kann viele Problemen für kommend Iterationen des Projektes verursachen, da eigentlich der Kontroller nichts von den UI wissen soll und es nur ein einziger Kontroller geben soll. Zum Beispiel wenn wir eine andere Art von UI erstellen wollen, vielleicht ein Menu im Terminal, oder eine dynamisch erzeugte WEB-Seite, dann treten hier viele Problemen auf, da wir zuerst den Kontroller umschreiben müssen, sodass es universell für allen Arten von UI's funktioniert.

Eventualitätsplanung

Dass heißt, dass wir in die zukünftige Iteration die Logik von dem View-Ebene auftrennen müssen, und die anderen 3 Kontroller zu einem einzigen machen, damit die Logik des Programmes klar befolget werden kann, ohne dass wir in der View-Ebene überprüfen müssen, wie Teilen einer Kontroller miteinander arbeiten.

Risiko Minimierung

Vom Anfang an, wenn man den MVC Designmuster befolgen will, sollten wir klar gendanken machen wie wir die Logik von dem Repository und dem UI abtrennen.

KAPITEL 12

Glossar

12.1. Einführung

Die folgende Liste von Begriffen beinhaltet deutsche Begriffe die häufig in den Universitäten benutzt werden und zu unserem Projekt relevant sind.

Was sind Kurse?

Ein Kurs ist eine zusammengehörende Folge von Unterrichtsstunden wo die Kenntnisse der Studierenden durch eine Prüfung evaluiert werden

Was sind Studenten?

Ein Student ist jemand, der an einer Hochschule studiert also hat somit, mindestens ein Lyzeum Abschluss und der vielleicht auch ein Abiturprüfung abgelegt hat

Was sind Professoren?

Ein Professor ist ein Träger eines Professorentitels, mehr dazu für unser Projekt ist ein Professor ein Hochschullehrer also jemand mit höchstem akademischem Titel

12.2. Ausdrücke und Begriffe

An einem Kurs angemeldet zu sein = an einem Kurs teil zu machen und am Ende oder während des Kurses eine Prüfung zu bestehen

Universitätskredite = sind Punkte die Studenten sammeln indem sie Prüfungen an Kurse bestehen

Maximale Einschreibungszahl = ist die maximale Anzahl von Studenten die an einem Kurs teilnehmen können

ID Nummer für Studenten/Professoren = ist ein einzigartiges Identifikationsnummer für eine Person