

# Duale Hochschule Baden-Württemberg Mannheim

## **Projekt II**

### **Studiengang Wirtschaftsinformatik**

Studienrichtung Software Engineering

**Gruppe Denver**

**Nico Himmelein, Nele Ecker, Lukas Freitag, Lars Langhammer,**

**Mireille Puschmann, Thomas Richter, Philipp Thiele,**

**Philip Wagner und Jan Vögeli**

**Projekt Colorado**

# Inhaltsverzeichnis

1 Aufgabenstellung.....	1
2 Eclipse Projekt.....	2
3 Architektur Modell.....	3
3.1 Architektur .....	3
3.2 Klassendiagramm .....	3
3.2.1 Models.....	3
3.2.2 Controller.....	7
3.2.3 Services .....	8
3.3 Allgemein .....	10
3.3.1 Docker .....	10
3.4 Backend.....	10
3.4.1 PostgreSQL.....	10
3.4.2 Spring.....	10
3.4.3 Tomcat .....	11
3.4.4 Hibernate.....	11
3.4.5 HATEOAS.....	11
3.5 Frontend.....	11
3.5.1 ReactJS.....	11
3.5.2 Material UI.....	11
3.5.3 Node.js .....	12
3.5.4 Express.js .....	12
3.5.5 OpenSSL.....	12
4 ER Diagramm.....	13
4.1 ER Diagramm PostgreSQL.....	13
4.2 ER Diagramm Hibernate .....	16
5 REST Schnittstellen.....	18
6 Autorisierung und Authentifizierung.....	19
7 Passwortschutz .....	20
8 Informationen zu der Anwendung.....	21
8.1 Fähigkeiten der Anwendung .....	21
8.2 Nutzung der Anwendung.....	21

# Abbildungsverzeichnis

Abbildung 1: Ordnerstruktur in Eclipse .....	2
Abbildung 2: Architektur des Projektes .....	3
Abbildung 3: Klassendiagramm Model .....	5
Abbildung 4: Klassendiagramme mit Methoden und Attributen .....	5
Abbildung 5: Klassendiagramme mit Methoden und Attribute .....	6
Abbildung 6: Klassendiagramm Controller .....	7
Abbildung 7: Klassendiagramm der Controller .....	7
Abbildung 8: Klassendiagramm der Controller .....	8
Abbildung 9: Klassendiagramm des HibernateControllers .....	8
Abbildung 10: Klassendiagramm Services .....	9
Abbildung 11: Klassendiagramm Services .....	9
Abbildung 12: ER-Diagramm mit allen Relationen .....	13
Abbildung 13: Relation course, course_lecture und course_denveruser .....	14
Abbildung 14: Relation userdenver .....	14
Abbildung 15: Relationen lecture, lecture_exercies, lectures_users .....	15
Abbildung 16: Relation exercise .....	15
Abbildung 17: Relationen role, userdenver_solution, users_roles .....	16
Abbildung 18: Relation solution .....	16
Abbildung 19: ER Diagramm der Hibernate Entitäten .....	17

# 1 Aufgabenstellung

Ziel dieses Projektes ist es eine Lernplattform zu entwickeln, die die Vorlesung Programmieren I unterstützen soll. Hierfür sind folgende Use Cases erstellt:

Der Dozent soll Aufgaben und Lösungswege erstellen können.

- Es soll möglich sein Kurse zu verwalten und diesen Aufgaben zuordnen zu können.
- Die Aufgaben sollen automatisch evaluiert werden können und online lösbar sein.
- Der Dozent soll eine Liste mit Auswertungen der Ergebnisse und eine Liste mit Aufgaben die von den Studenten bearbeitet wurden einsehen können.
- Es sollen Programmieraufgaben für die Sprachen Java und JavaScript von dem Dozenten erstellt werden können.

Für dieses Projekt wurden folgende Anforderungen spezifiziert:

- Das Projekt soll in einem docker-compose File vorliegen.
- Es soll ein Authentifizierungs- und Autorisierungskonzept erstellt werden.
- Die Kommunikation zwischen dem Client und einem auf Java basierenden Server soll mit SSL verschlüsselt werden und mit REST-Interfaces erfolgen.
- Die Benutzeroberfläche soll mit der JavaScript-Bibliothek ReactJS erstellt werden. Zudem soll in der Oberfläche das Syntax-Highlighting in Editoren vorhanden sein.
- In der Oberfläche soll es möglich sein Kurse und Studenten zu verwalten. Der Dozent soll Kurse und Studenten verwalten können. Zudem soll eine Profilverwaltung für Student und Dozent eingerichtet werden. User in diesem System ist entweder der Student oder der Dozent. Ein Kurs kann mehrere Dozenten zugeteilt bekommen. Die Aufgaben sollen an mehrere Kurse zugewiesen werden können.
- Der Dozent soll mindestens die letzte Lösung des Studenten einsehen können.
- Die Datenbank soll PostgreSQL sein.

**Link zum Repository: <https://github.com/yaumo/Denver>**

## 2 Eclipse Projekt

Um den Aufbau des Projektes verständlicher zu machen, wird in diesem Kapitel der Aufbau des Eclipse Projektes erläutert.

Abbildung 1 zeigt das Eclipse Projekt.

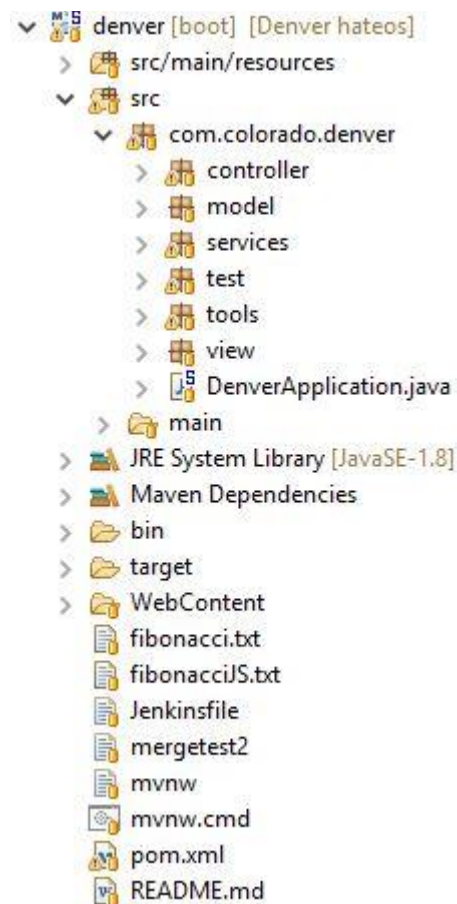


Abbildung 1: Ordnerstruktur in Eclipse

In dem Ordner `src/main/resources` sind Konfigurationsdateien enthalten. Die Konfigurationsdateien sind:

- `Application.properties`:
- `Hibernate.cfg.xml`

Im `/src` Ordner befindet sich die Backend Logik der Webanwendung in Java.

In dem Ordner `WebContent` ist die Webapp vorhanden basierend auf Node.js und React.

## 3 Architektur Modell

### 3.1 Architektur

Die Architektur lässt sich grob in die Datenbank, das Backend, den Expressserver mit Frontend und den Browser unterteilen.

Die Datenbank kommuniziert durch Port 5432 mit dem Backend.

Die Kommunikation zwischen dem Backend und dem Frontend erfolgt auf Port 8443 und wird mit SSL verschlüsselt. Das Frontend wird durch den Express Server ausgeliefert. Die Navigation erfolgt durch die React-App.

Auch eine verschlüsselte Kommunikation mit SSL zwischen dem Express Server und dem Browser ist vorhanden. Die Kommunikation erfolgt auf Port 8081.

Abbildung 2 zeigt die beschriebene Architektur nochmals auf.

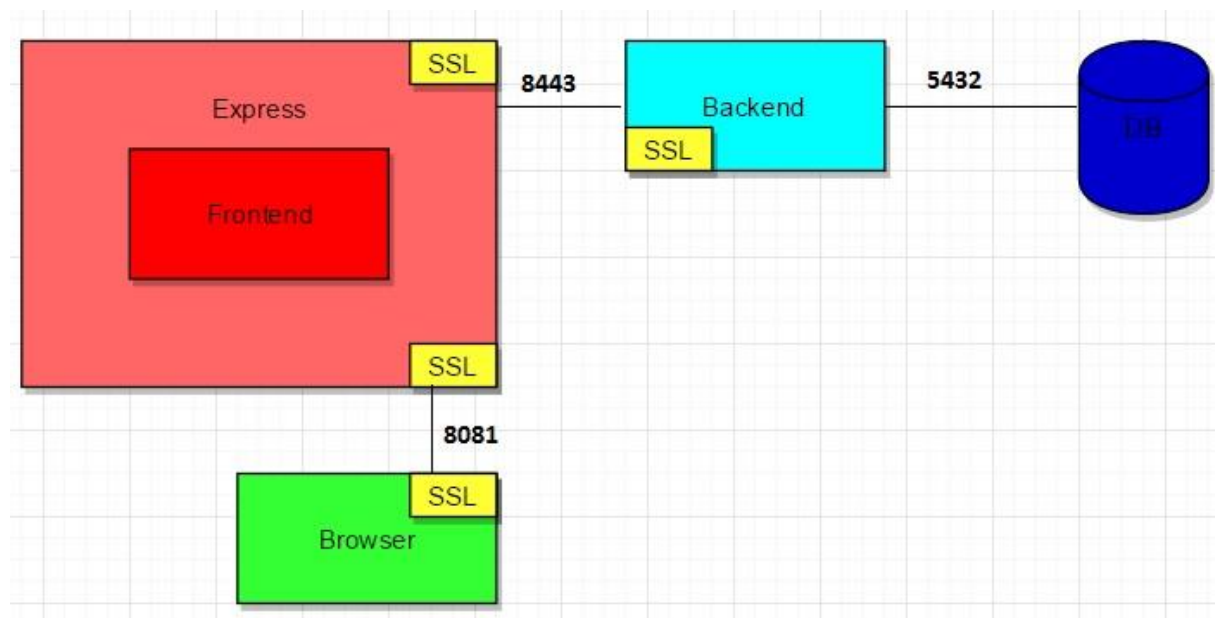


Abbildung 2: Architektur des Projektes

### 3.2 Klassendiagramm

Die Klassendiagramme teilen sich in Models, Services und Controller.

#### 3.2.1 Models

Als Superklassen beschreiben BaseEntity und EducationEntity Eigenschaften, welche vererbte Entities ebenfalls besitzen. Dadurch werden sie nochmals abstrahiert, aber nicht persistiert. Beispielsweise besitzt jedes Entity eine Hibernate-ID. Aber diese muss nicht in jeder Entity Klasse nochmals definiert werden.

Es wird nur eine Userklasse benötigt weil die Berechtigungen mit dem Attribut ‚principal‘ identifiziert werden.

Für die eigentliche Aufgabenstellung der Programmieraufgaben existiert das Entity ‚exercise‘. Für die Lösungen der Studenten ist das Entity ‚solution‘ gedacht. Dadurch kann eine Aufgabe mehrfach in mehreren Jahrgängen verwendet werden.

In Abbildung 3 wird das Klassendiagramm des Models gezeigt.

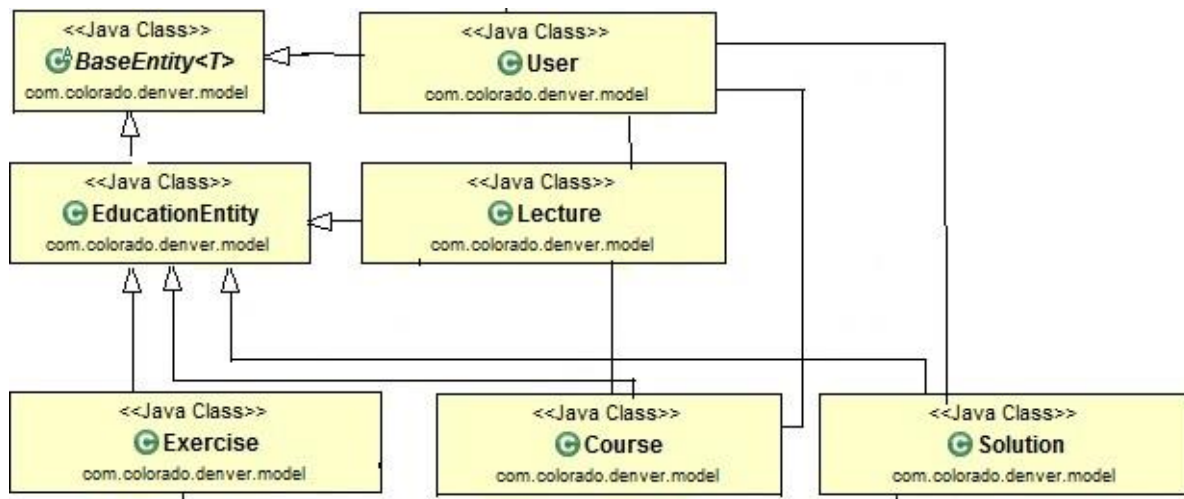


Abbildung 3: Klassendiagramm Model

In Abbildung 4 und 5 werden die Klassendiagramme mit Attributen und Methoden dargestellt.

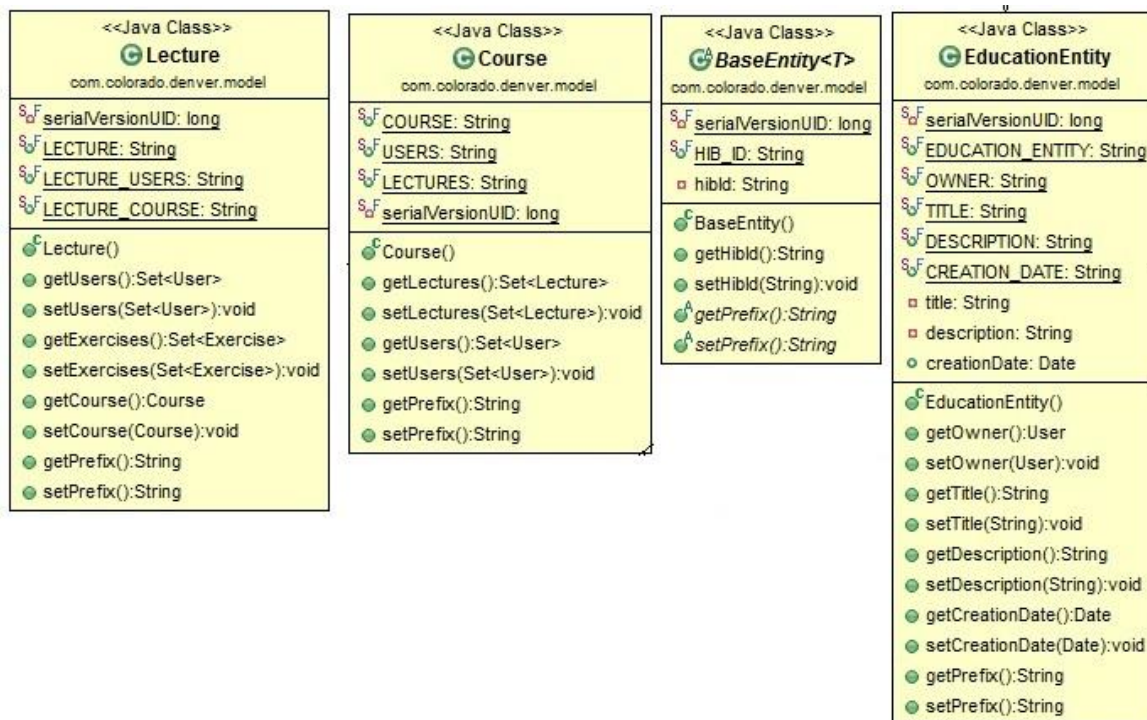


Abbildung 4: Klassendiagramme mit Methoden und Attributen





Abbildung 5: Klassendiagramme mit Methoden und Attribute

### 3.2.2 Controller

Es existieren für fast alle Entities passende Controller, welche die Strukturen der jeweiligen Entities gesondert behandeln können. Abbildung 6 zeigt die Controller.

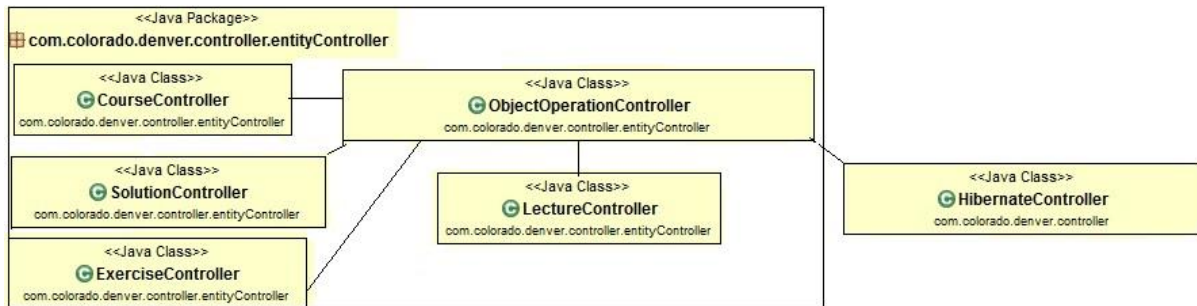


Abbildung 6: Klassendiagramm Controller

In den Abbildungen 7 bis 9 werden die Controller mit Attributen und Controllern dargestellt.

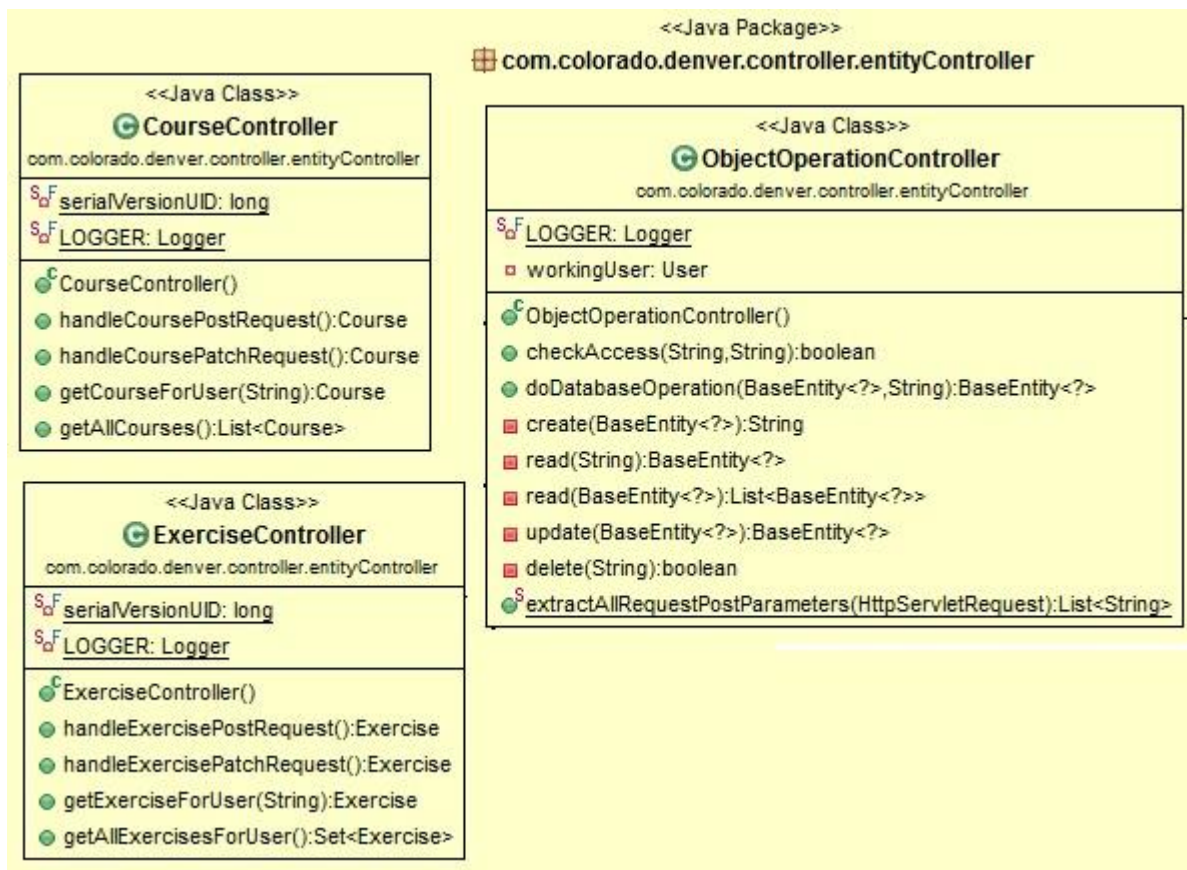


Abbildung 7: Klassendiagramm der Controller

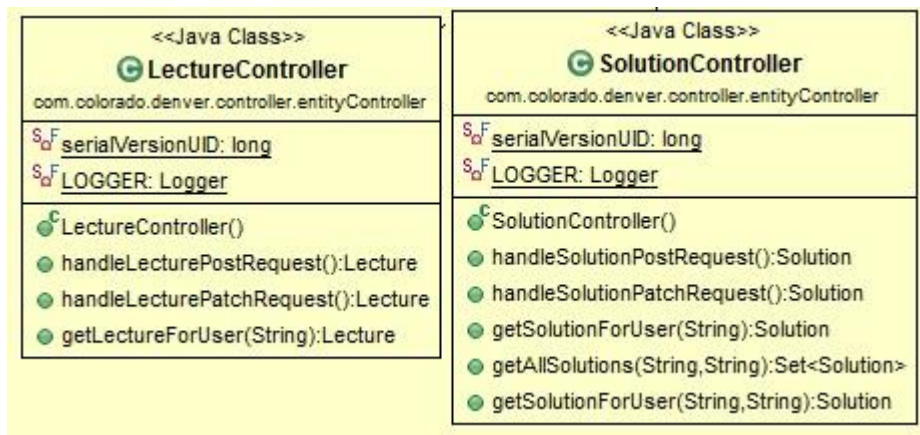


Abbildung 8: Klassendiagramm der Controller

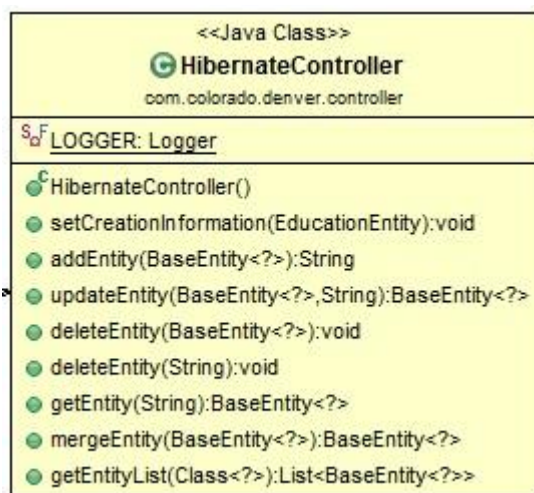


Abbildung 9: Klassendiagramm des HibernateControllers

### 3.2.3 Services

Services dienen für die Schnittstellen zwischen der Controller und Objekte auf diesen sie arbeiten. In den Abbildungen 10 und 11 werden die Services gezeigt.



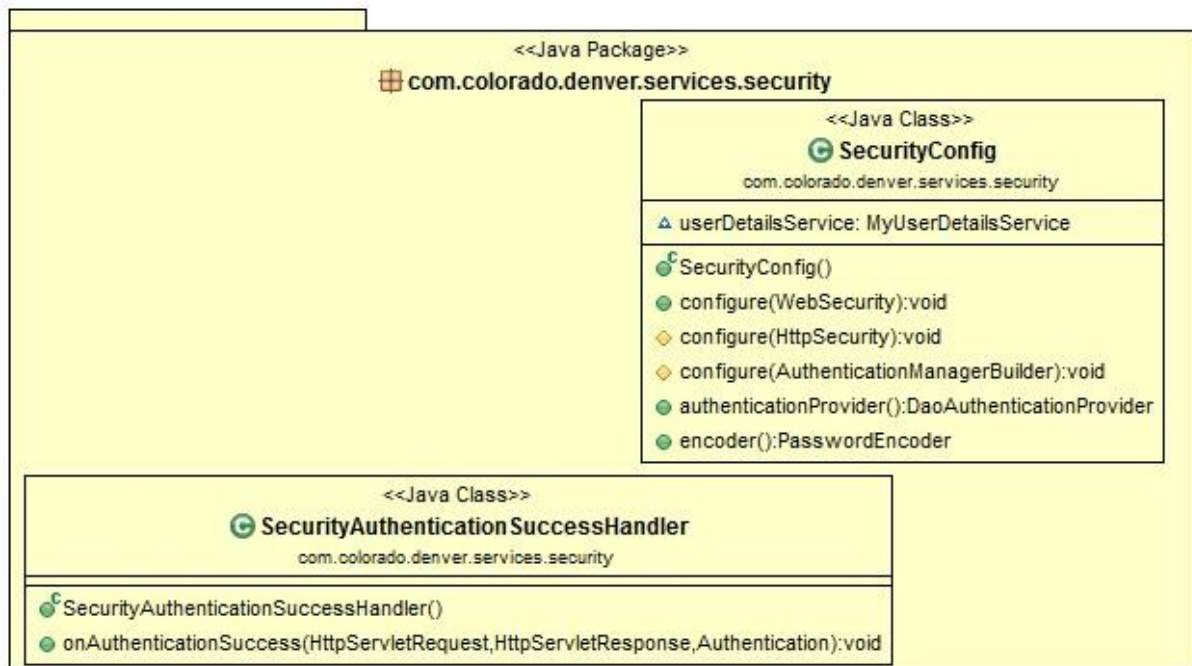


Abbildung 10: Klassendiagramm Services

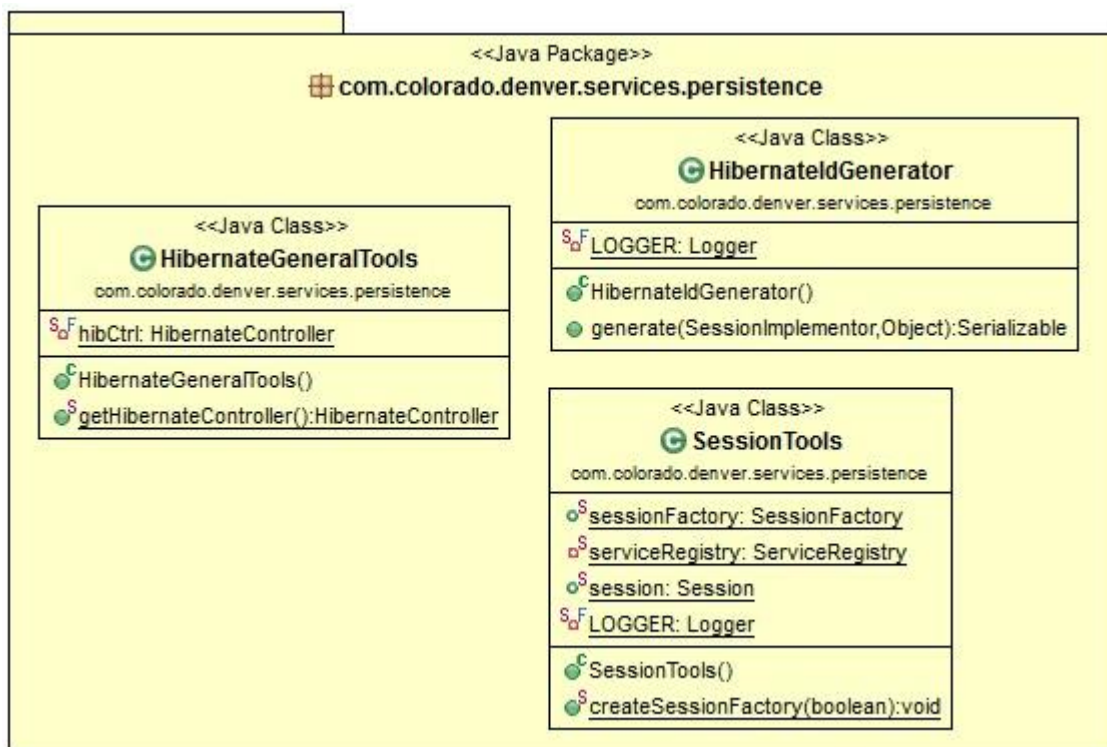


Abbildung 11: Klassendiagramm Services

## **3.3 Allgemein**

### **3.3.1 Docker**

Docker ist eine Opensource-Software. Mit Docker ist es möglich Anwendungen mit einer Betriebssystemvisualisierung in Containern zu isolieren. Es vereinfacht die Bereitstellung von Komponenten. Dateien können einfach transportiert und installiert werden. Docker ermöglicht die Trennung von Rechnern und Ressourcen und basiert auf Linux.

Docker wird in diesem Projekt dazu verwendet eine Pipeline mit allen benötigten Ressourcen aufzubauen.

## **3.4 Backend**

### **3.4.1 PostgreSQL**

PostgreSQL ist ein freies relationales Datenbankmanagementsystem, welches ACID konform ist. Es unterstützt Datentypen, Operationen, Aggregate und Funktionen. In PostgreSQL wird es ermöglicht Abfragen mit Unterabfragen zu bilden.

In diesem Projekt wird PostgreSQL für die Persistierung der Daten eingesetzt.

### **3.4.2 Spring**

Spring ist ein quelloffenes Framework für Java-Plattformen. Durch Spring wird die Entwicklung in Java vereinfacht. Objekten werden benötigte Ressourcen zugewiesen. Technische Aspekte wie Transaktionen oder Sicherheit können isoliert werden. Zudem ist die Vereinfachung von APIS durch Spring möglich.

In diesem Projekt wurden hauptsächlich Spring Security und Spring Boot eingesetzt.

#### **3.4.2.1 Spring Security**

Spring Security dient zur Absicherung von Java-Anwendungen und Webseiten und beinhaltet Login/Logout/Registrierung sowie Session Management und Schutz vor fremden Zugriff.

#### **3.4.2.2 Spring Boot**

Spring Boot ist ein Framework für die einfache Entwicklung eigenständig lauffähiger Spring-Anwendungen per Konvention vor der Konfiguration. Die Konfiguration erfolgt ohne XML.

### **3.4.3 Tomcat**

Tomcat ist ein von Apache entwickelter auf Java basierender Opensource Webserver und Webcontainer, welcher speziell für Java Servlets und Java Server Pages eingesetzt wird.

In diesem Projekt wird Tomcat als Server eingesetzt, um die Kommunikation zu Clients herstellen zu können.

### **3.4.4 Hibernate**

Hibernate bildet ein Opensource-Persistenz und ORM-Framework für Java ab. Es ermöglicht die objektorientierte Abbildung von Datenbankentitäten. Objekte mit Attributen und Methoden werden in relationalen Datenbanken gespeichert um aus Datensätzen Objekte zu erzeugen. Auch die Abbildung zwischen Objekten ist möglich. Datenbankzugriffe mit SQL-Anweisungen werden in einem SQL-ähnlichen Dialekt generiert.

### **3.4.5 HATEOAS**

HATEOAS (Hypermedia As The Engine Of Application State) bildet die REST Anforderungen der Anwendung. Der Client arbeitet mit einer Netzwerkanwendung und basierend auf Entity-Informationen. HATEOAS benötigt kein Wissen über die Applikation oder wie die Applikation arbeitet.

## **3.5 Frontend**

### **3.5.1 ReactJS**

ReactJS ist eine Javascript-Bibliothek, die von Facebook entwickelt wurde. Mit ReactJS ist es möglich Webanwendungen zu erstellen. Es stellt ein Grundgerüst für die Ausgabe von User-Interface Komponenten in HTML zur Verfügung. Der Aufbau ist hierarchisch und wird als selbstdefinierte HTML-Tags repräsentiert. Mittels NodeJS wird der Code serverseitig gerendert. ReactJS nutzt nicht das strikte MVC Konzept.

In diesem Projekt wird ReactJS für die Entwicklung der grafischen Oberfläche eingesetzt.

### **3.5.2 Material UI**

Material UI ist ähnlich zu Material Design von Google und für ReactJS entwickelt. Die Customization teilt sich in themes, styles und colors. Dieses Design ist von Google entwickelt worden und stellt diverse Komponenten wie Buttons oder Textfelder vorformatiert zur Verfügung. Es trägt dazu bei, dass das UI einheitlich aussieht und intuitiv zu bedienen ist.

In diesem Projekt wird Material UI für das Design der Benutzeroberfläche verwendet.

Es wurden Komponenten erstellt die in jedem View wiederverwendet werden können, wie zum Beispiel die Komponente AceEditor, die den Texteditor beinhaltet. Dieses Konzept entspricht dem Aufbau und den Vorgaben von ReactJS. Es wurde sich an der Farbgebung an dem alten Design von Moodle orientiert, da Studenten mit der Oberfläche vertraut sind.

### **3.5.3 Node.js**

Node.js ist eine serverseitige Plattform, die den Betrieb von Netzerkanwendungen ermöglicht. Mit Node.js ist es möglich Webserver zu realisieren. Es wird in einer JavaScript-Laufzeitumgebung ausgeführt. JavaScript gibt eine ereignisgesteuerte Architektur vor. Zudem besitzt Node.js Module, welche als Ergänzungen angesehen werden können. Einige Module können direkt in das Binärpaket kompiliert werden. Module können den asynchronen Netzwerkzugriff, Adapter, Puffer, Zeitgeber und allgemein-gehaltene Datenstrom-Klassen darstellen. Zur Verwaltung der Module gibt es einen Paketmanager npm.

Node.js wird in diesem Projekt als Webserver für die React-Anwendung verwendet.

### **3.5.4 Express.js**

Express.js ist ein schnelles, offenes und unkompliziertes Web-Framework. Es basiert auf Node.js. Mit Express.js ist es möglich schnell lauffähige APIS zu erstellen. Es ist ein Standard Server Framework für Node.js.

In diesem Projekt wird Express.js als Reverseserver mit SSL-Anwendung verwendet.

SSL wurde mit OpenSSL umgesetzt.

### **3.5.5 OpenSSL**

OpenSSL ist eine freie Software für Transport Layer Security. Sie umfasst die Implementierung der Netzwerkprotokolle und verschiedene Verschlüsselungen.

Es gibt ein Programm openssl für die Kommandozeile, das es ermöglicht Zertifikate zu beantragen, zu erzeugen und zu verwalten.

Eine Basisbibliothek basierend auf C stellt allgemeine Funktionen zum ver- und entschlüsseln und diverse weitere Werkzeuge bereit.

OpenSSL wurde eingesetzt um ein Zertifikat und eine Key zu erstellen.

## 4 ER Diagramm

Die ER Diagramme von Hibernate und PostgreSQL unterscheiden sich. In dieser Dokumentation werden beide ER Diagramme aufgezeigt.

### 4.1 ER Diagramm PostgreSQL

Für die Relationen in PostgreSQL mussten mehrere Tabellen, die nur Fremdschlüssel enthalten, erzeugt werden. Insgesamt wurden zwölf Tabellen erzeugt. Alle Relationen, die nicht nur Fremdschlüssel enthalten, besitzen einen Primärschlüssel auf der Entität hibid.

In Abbildung 12 werden alle Relationen und deren Fremdschlüsselbeziehungen aufgezeigt.

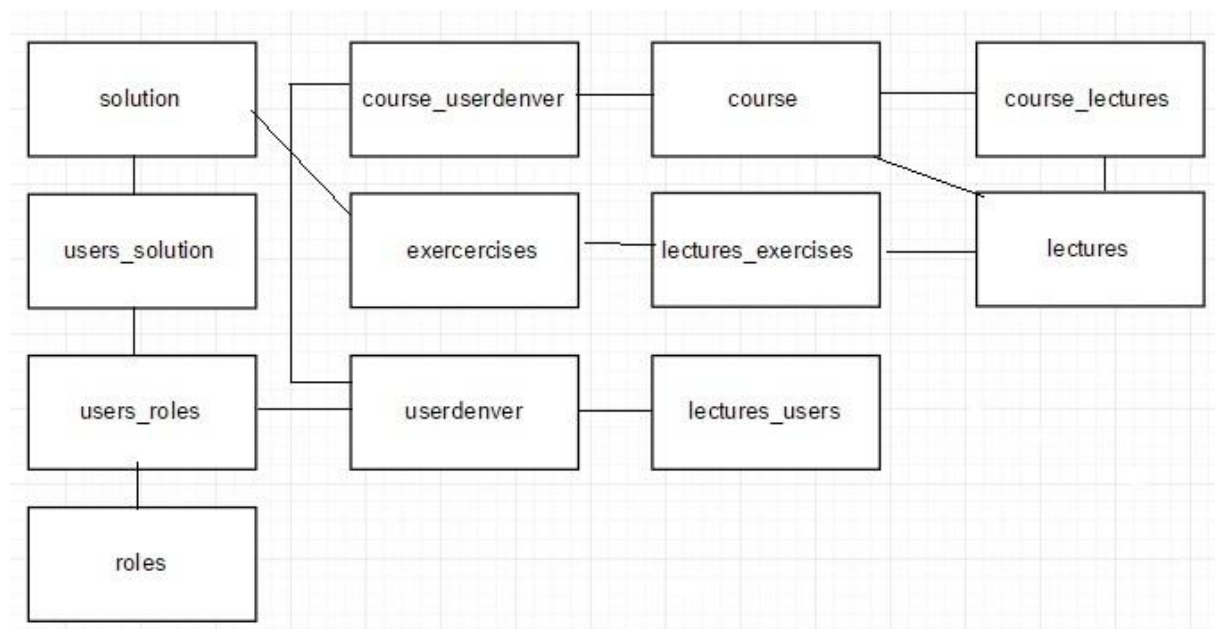


Abbildung 12: ER-Diagramm mit allen Relationen



Abbildung 13 zeigt die Relation ‚course‘, welche die Kurse der Hochschule darstellen. Zudem werden in dieser Abbildung die Relationen ‚course\_lecture‘ und ‚course\_userdenver‘ welche reine Fremdschlüssel-Relationen für die Relationen ‚course‘, ‚lecture‘ und ‚userdenver‘ darstellen abgebildet.

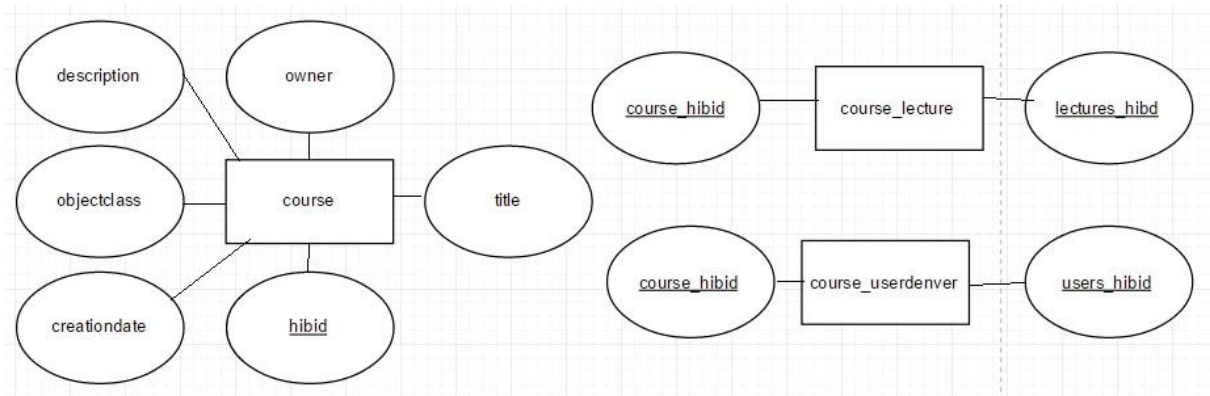


Abbildung 13: Relation course, course\_lecture und course\_denveruser

In Abbildung 14 wird die Relation ‚userdenver‘ abgebildet, welche den Benutzer darstellt.

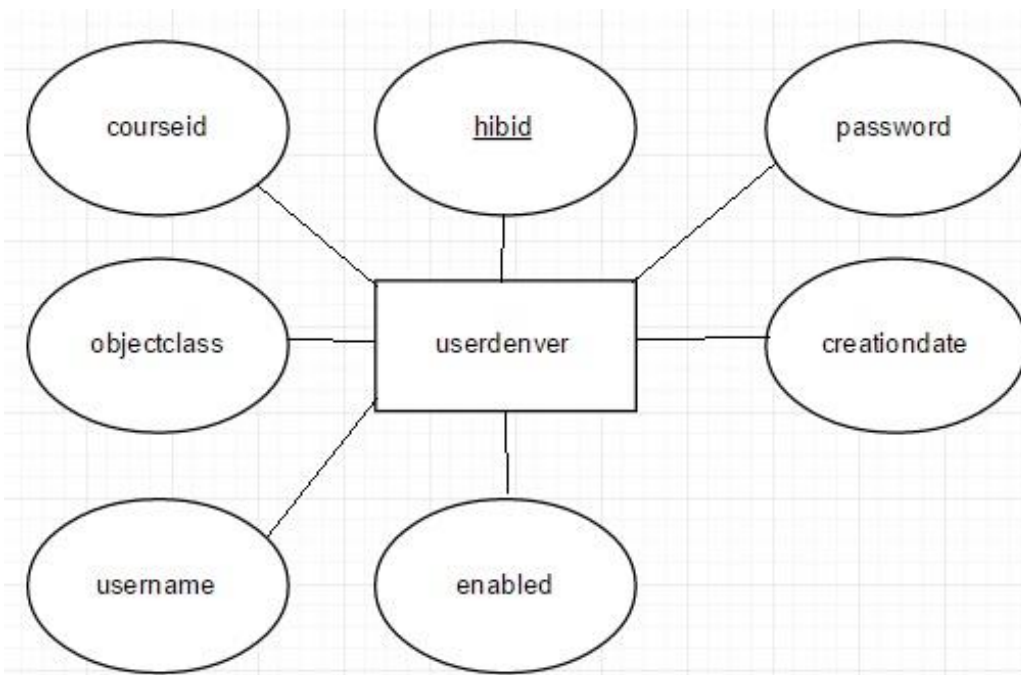


Abbildung 14: Relation userdenver

Die Relationen ‚lecture‘, ‚lectures\_exercises‘ und ‚lectures\_user‘ werden in Abbildung 15 dargestellt. Die Relationen ‚lecture\_exercises‘ und ‚lectures\_user‘ bilden reine Fremdschlüsselbeziehungen ab.

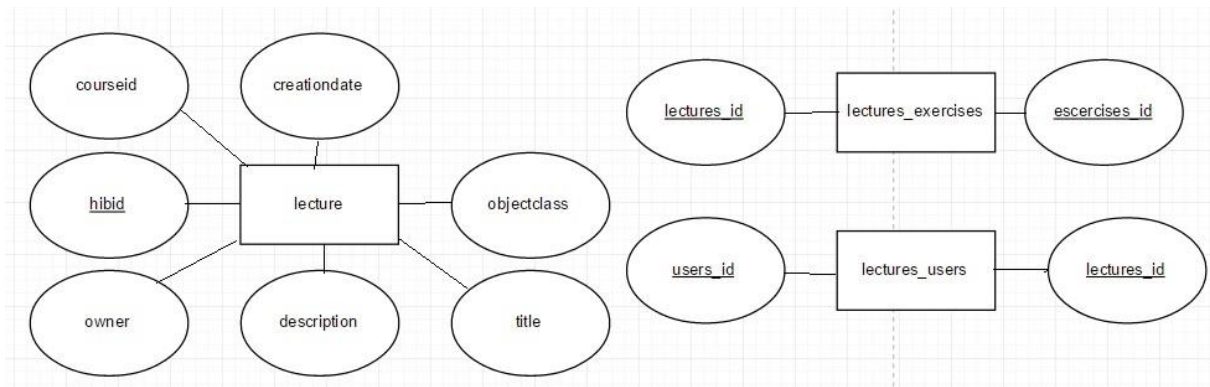


Abbildung 15: Relationen lecture, lecture\_exercises, lectures\_users

In Abbildung 16 wird die Relation 'exercise' dargestellt. Sie bildet die Aufgabe für einen Studenten ab.

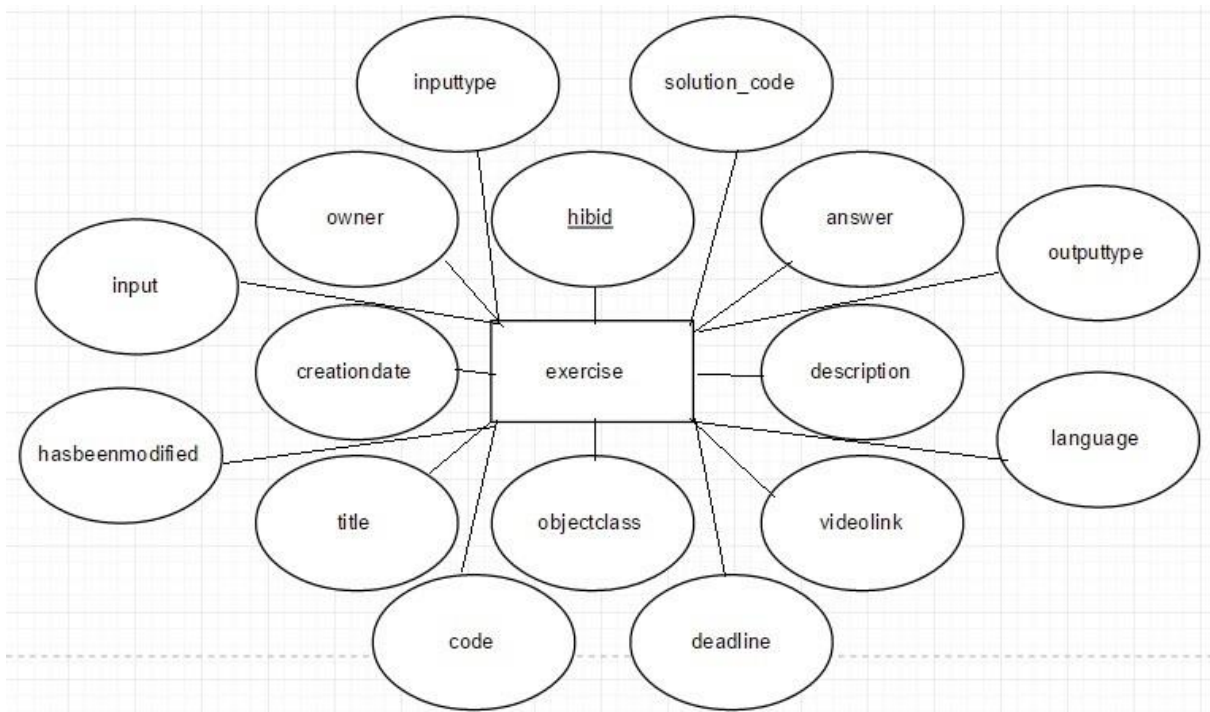


Abbildung 16: Relation exercise

Die Relationen ‚role‘, ‚userdenver\_solution‘ und ‚users\_roles‘ werden in Abbildung 17 dargestellt. In der Relation ‚role‘ sind nur die Rollen Student und Dozent vorhanden. Die Relationen ‚userdenver\_solution‘ und ‚users\_roles‘ werden reine Fremdschlüsselbeziehungen abgebildet.

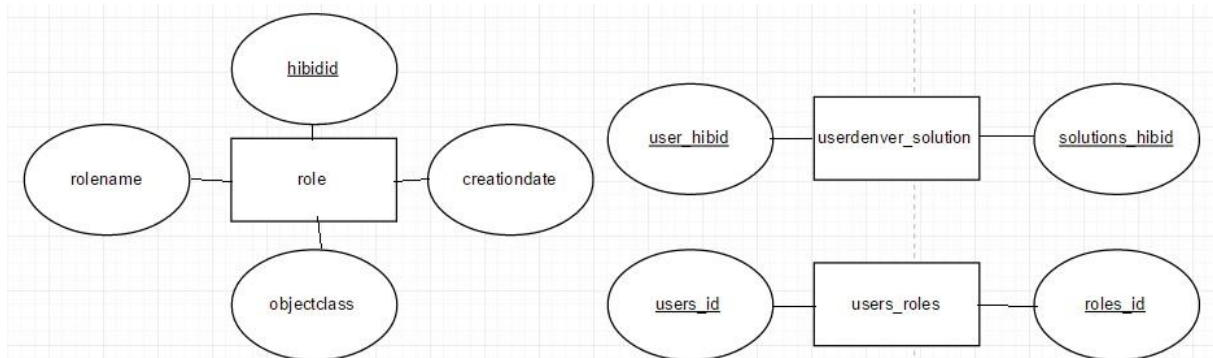


Abbildung 17: Relationen role, userdenver\_solution, users\_roles

Abbildung 18 bildet die Relation ‚solution‘ ab. Diese stellt die Lösung des Dozenten und die Lösung des Studenten dar.

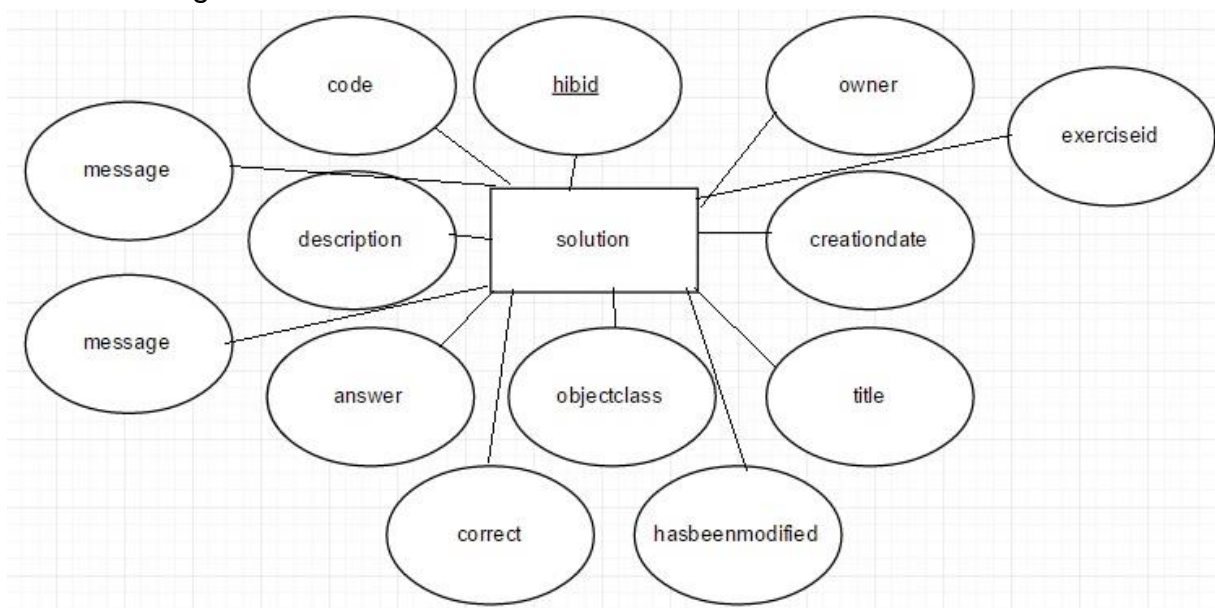


Abbildung 18: Relation solution

## 4.2 ER Diagramm Hibernate

Das ER Diagramm der Entitäten von Hibernate weicht zum Teil von dem ER Diagramm der Relationen von PostgreSQL sehr ab.

Abbildung 19 verdeutlicht diese Unterschiede.

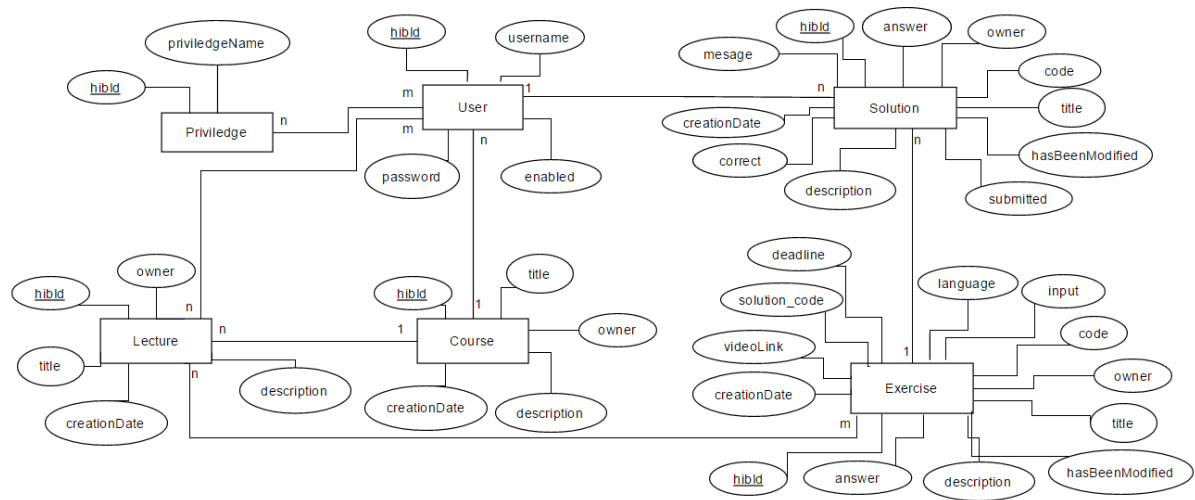


Abbildung 19: ER Diagramm der Hibernate Entitäten

Zudem sind in Hibernate keine Entitäten vorhanden, die nur Fremdschlüsselbeziehungen abbilden, vorhanden.

## 5 REST Schnittstellen

Die Kommunikation mit dem Backend erfolgt durch asynchrone AJAX Calls mit den http-Methoden POST, GET, PATCH und DELETE. Diese Anfragen werden an den Reverse Proxy Express geschickt, mit den nötigen HTTP-Headern versehen und an das Backend weitergeleitet. Die Header sind aufgrund der Cross Origin Policy nötig, da beispielsweise Server A ohne Header nicht Server B ansprechen kann.

## 6 Autorisierung und Authentifizierung

Um die Autorisierung und die Authentifizierung realisieren zu können wurde ein Express Server benötigt. Express wird für die Authentifizierung als Reverseserver mit SSL eingesetzt.

Für das Login werden der User und die Passwortabfrage benötigt. Die Überprüfung dieser beiden Informationen erfolgt über die Datenbank. Wenn der User nicht vorhanden ist oder das Passwort falsch ist erscheint eine Fehlermeldung.

Wenn der User in der Datenbank existiert und das Passwort korrekt ist erfolgt das Login. Hierfür werden nun eine Session und ein Cookie erstellt.

Die Überprüfung des Passwortes läuft wie folgt ab:

Das Passwort wird eingetippt und eine Login-Abfrage wird gestartet. Das gesendete Passwort wird verschlüsselt und an die Datenbank gesendet. Die Datenbank vergleicht das verschlüsselte Passwort mit dem gespeicherten verschlüsselten Passwort für den entsprechenden User. Wenn die Passwörter übereinstimmen erfolgt der Login.

Für die Session ist zu beachten, dass ein Sessioncookie erstellt wird. Wenn der Benutzer die Seite wechselt wird die Überprüfung ob eine Session bereits besteht durch einen Call der die Session ID übermittelt erfolgen. Die Session wird in der Datenbank für den Benutzer eingetragen und die Überprüfung ob der Benutzer für diese Seite berechtigt ist erfolgt. Mit dem Logout wird die Session beendet und die Session ID wird aus der Datenbank rausgelöscht.

## 7 Passwortschutz

Der Passwortschutz findet wie folgt statt:

Das Passwort wird bei der Registrierung in die Datenbank verschlüsselt gespeichert. Die Ver- und Entschlüsselung der Passwörter erfolgt über die Java-Klasse Bcrypt.

Beim Login wird das Passwort in der Loginseite eingetragen. Wenn eine Loginanfrage abgeschickt wird, wird das eingegebene Passwort mit `bcrypt.hashpw()` verschlüsselt.

Der Vergleich der Passwörter aus der Datenbank und der Loginseite erfolgt mit der Methode `bcrypt.checkpw()`.

Bcrypt generiert eine 128 Bit langen Salt für die Verschlüsselung.

## 8 Informationen zu der Anwendung

### 8.1 Fähigkeiten der Anwendung

Der Student kann Aufgaben, die seinem Kurs zugeteilt sind bearbeiten. Er kann Lösungen abgegeben, wobei es möglich mehrere Lösungen ab zu geben. Bei dem Lösen der Aufgaben wird der Student durch eine Ace Editor durch Syntax-Highlighting und Autovervollständigung unterstützt. Auch kann der Student seine Lösung überprüfen lassen. Das Bearbeiten von abgegebenen Aufgaben ist nicht möglich. Es wird immer nur die letzte abgegebene Aufgabe als Lösung gespeichert. Dem Studenten ist es möglich sein Profil anzuschauen und sein Passwort zu ändern.

Auch dem Dozenten wird die Profilverwaltung ermöglicht. Der Dozent kann Aufgaben erstellen, Aufgaben Kurse zuteilen und Lösungen erstellen. Zudem wird dem Dozenten eine Liste mit den abgegebenen Aufgaben und mit den zugeteilten Aufgaben mit Status der Aufgabe angezeigt. Der Dozent kann auswählen ob die Aufgaben für Java oder JavaScript erstellt wurde. Zusätzlich ist es möglich eine Aufgabe mit einer Deadline zu versehen. Die Aufgaben können auch einem Tutor zugeteilt werden.

### 8.2 Nutzung der Anwendung

Eine wichtige Voraussetzung zur Verwendung der Anwendung ist, dass die IP-Adresse des Docker Hosts auf 192.168.99.100 liegt und somit alle Container unter dieser IP-Adresse aufrufbar sind, wenn ihre Ports geöffnet sind.

Bevor die Docker-compose Datei ausgeführt wird muss in dem Docker Terminal zu der `entrypoint.sh` in dem `Dockerjava` Ordner hin navigiert werden. Die `entrypoint.sh` muss mit dem `dos2unix` Befehl kompiliert werden.

Durch das Ausführen der Docker-compose Datei wird Node.js lokal auf dem Rechner installiert. Auch wird PostgreSQL lokal installiert. Danach werden die Docker für das Backend und das Frontend erzeugt. Die Installation der benötigten Software dauert etwa fünf bis zehn Minuten.

Danach kann in dem Browser unter <https://192.168.99.100:8081> das Frontend geöffnet werden.