Pflichtenheft

Gruppe 02

15. März 2015

Inhaltsverzeichnis

1 Einführung **3**
- 1.1 Überblick . . . 3
- 1.2 Technologien . . . 3

2 System Übersicht **5**
- 2.1 Komponentendiagramm . . . 5
- 2.2 Verteilungsdiagramm . . . 6
  - 2.2.1 Android Device . . . 6
  - 2.2.2 Client Computer . . . 6
  - 2.2.3 Server . . . 6
- 2.3 Paketdiagramm . . . 7
  - 2.3.1 Grails Anwendung . . . 7
  - 2.3.2 Android App . . . 7

3 Ordner Struktur **8**

4 Klassen **9**
- 4.1 Webseite . . . 9
  - 4.1.1 Modellklassen . . . 9
  - 4.1.2 Controllerklassen . . . 11
  - 4.1.3 Serviceklassen . . . 13
- 4.2 Android App . . . 15
  - 4.2.1 Activities . . . 16
  - 4.2.2 Adapters . . . 16
  - 4.2.3 Tasks . . . 17
  - 4.2.4 Utils . . . 18

5 Dynamisches Verhalten **19**
- 5.1 Login/Registrierung . . . 19
- 5.2 Aktivität erledigen . . . 20
- 5.3 Rangliste ansehen . . . 20
- 5.4 Team erstellen . . . 21
- 5.5 Team blocken . . . 22
- 5.6 Vorschlag erstellen . . . 23
- 5.7 Vorschlag bewerten . . . 24
- 5.8 Statistiken exportieren . . . 25
- 5.9 Vorschlag in Aktivität umwandeln . . . 26
- 5.10 App: Benutzerrangliste ansehen . . . 26

6 Anhang **27**
- 6.1 Klassen Diagramme . . . 27

1 Einführung

1.1 Überblick

Dieses Dokument stellt den Systementwurf des EnergyChallenge-Systems dar. Im Folgenden werden diese Themen behandelt:

- Kurze Einleitung in die verwendeten Technologien
- Beschreibung der beteiligten Komponenten und deren Verteilung auf Hardware
- Strukturierung des Systems in einzelne Pakete
- Aufbau und Beschreibung der in den Paketen enthaltenen Klassen
- Beschreibung des Verhaltens von einzelnen Anwendungsfällen mit Hilfe von Sequenzdiagrammen

1.2 Technologien

Das in diesem Dokument beschriebene System verwendet die folgenden Technologien: Android Android ist ein auf mobilen Geräten lauffähiges Betriebssystem sowie eine Software-Plattform für z.B. Smartphones und Tablets , welches auf einer stark abgeänderten Version von Linux basiert (es wird ein Linux-Kernel verwendet). Apps, also Android Applikationen, laufen in der Android Runtime, einer Android-eigenen Laufzeitumgebung (ab Android 5.0).

- http://www.android.com

Java Java ist die Programmiersprache, in der dieses Projekt größtenteils geschrieben ist. Dabei kommt Java meistens nicht in seiner „ursprünglichen“ Form vor, sondern wird z.B. durch Android Klassen ergänzt.

- http://java.oracle.com

GRAILS GRAILS ist ein auf der Programmiersprache Groovy aufbauendes Web Application Framework, welches mit relativ geringem Aufwand die Möglichkeit bietet, Webseiten mit Datenbank Anbindung zu erstellen. Mit GRAILS werden auch sogenannte Groovy Server Pages (GSP) erzeugt, welche Inhalte in sonst statische Webseiten einbinden können.

- http://grails.org

GORM GORM oder auch GRAILS Object-Relational Mapping ist GRAILS' integriertes ORM-Tool welches Hibernate 4 zum persistieren von Daten in einer Datenbank verwendet.

- http://grails.org/doc/2.3.x/guide/GORM.html
- http://www.hibernate.org

Apache Shiro Apache Shiro ist ein Java Security Framework, welches zur Autorisierung und Authentifizierung von Benutzern auf Webseiten verwendet wird. Es wird das Apache Shiro Plugin für Grails verwendet, welches viele Vorgänge automatisiert und vereinfacht.

- http://shiro.apache.org
- http://grails.org/plugin/shiro

2 System Übersicht

2.1 Komponentendiagramm

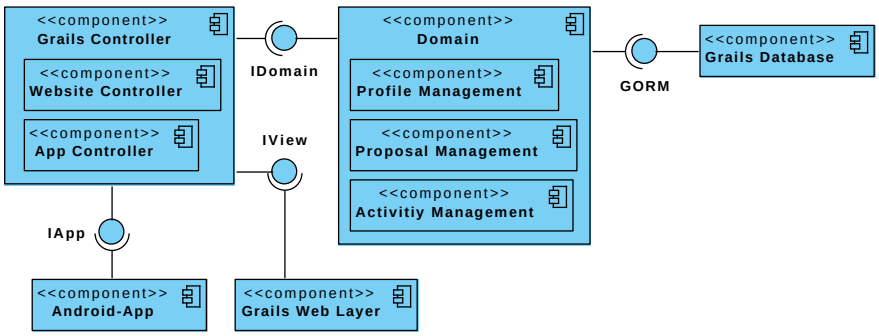

Abbildung 1: Komponentendiagramm

Grails Database Die Grails Database Komponente ist die zentrale Datenbank, die auf dem Server liegt und sämtliche Daten enthält, auf die, über das durch *Grail's Object Relational Mapping* (GORM) bereitgestellte Interface, zugegriffen werden kann.

Domain Die Domain Komponente ist die Modell Repräsentation der Daten in der Datenbank. Der Zugriff erfolgt über das IDomain Interface. Die Domain Komponenten ist unterteilt in die Subkomponenten *Profile Management*, *Proposal Management* und *Activity Management*. Das Profile Management ist zuständig für die Verwaltung der Modellrepräsentation von Team- und Benutzerprofilen. Das Proposal Managment ist zuständig für die Verwaltung der Modellrepräsentation von (Aktivitäts-) Vorschlägen. Die Komponente Activity Management ist zuständig für die Verwaltung der Modellrepräsentation von Aktivitäten.

Grails Web Layer Die Grails Web Layer Komponente kommuniziert mit dem Grails Controller über das IView Interface und stellt die GSP Dateien zur Verfügung, die das Layout der Webseite spezifizieren.

Grails Controller Die Controller Komponente besteht aus den einzelnen Controllern für Webseite und Android App und ist zuständig für die Verarbeitung sämtlicher Daten und Anfragen, die über die IApp, IDomain und IView Schnittstellen kommuniziert werden.

Android App Die Android App, dient in erster Linie zur Repräsentation der Daten vom Server auf Mobilgeräten. Sie kommuniziert über das IApp Interface mit dem Controller.

2.2 Verteilungsdiagramm

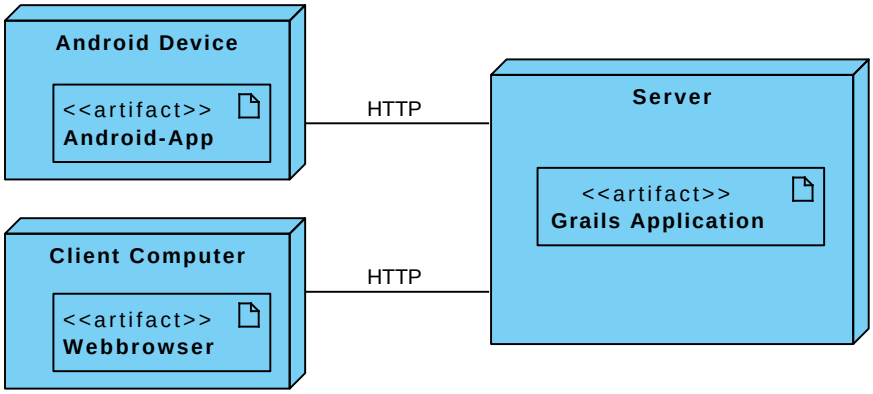

Abbildung 2: Verteilungsdiagramm

2.2.1 Android Device

Das Android Device ist das Arbeitswerkzeug für die Benutzer. Auf dem Gerät ist eine native Android-Applikation installiert, mit der Daten eingegeben und abgerufen werden können. Der Benutzer muss bereits registriert sein, um die Applikation nutzen zu können.

2.2.2 Client Computer

Benutzer können mit Hilfe eines Computers über einen Webbrowser auf eine Webseite zugreifen, welche, nach erfolgreichem einloggen, verschiedene Funktionen zur Aktivitäts- und Profilverwaltung anbietet.Des Weiteren können hier auch anonymisierte Statistiken angezeigt und diese statistischen Daten exportiert werden.

2.2.3 Server

Der Server stellt die eben erwähnte Webseite bereit. Hier werden gegebenenfalls Datenanfragen von den Android-Geräten verarbeitet.

2.3 Paketdiagramm

2.3.1 Grails Anwendung

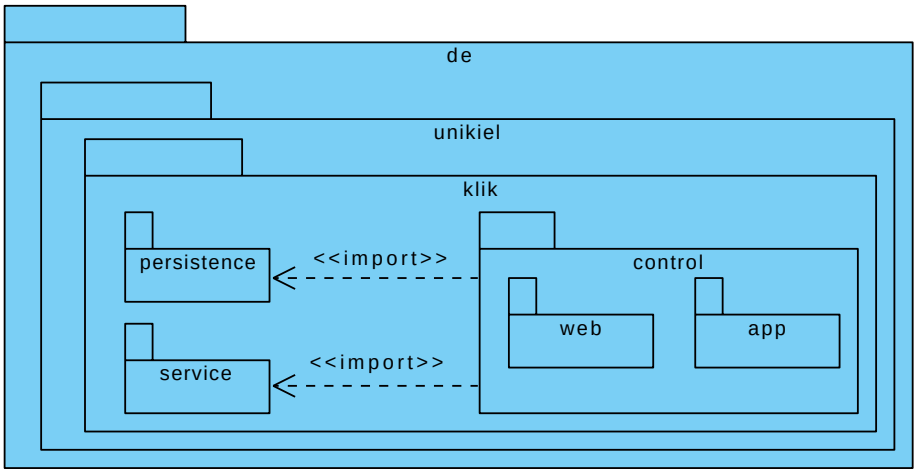

Abbildung 3: Pakete der Grails Anwendung

2.3.2 Android App

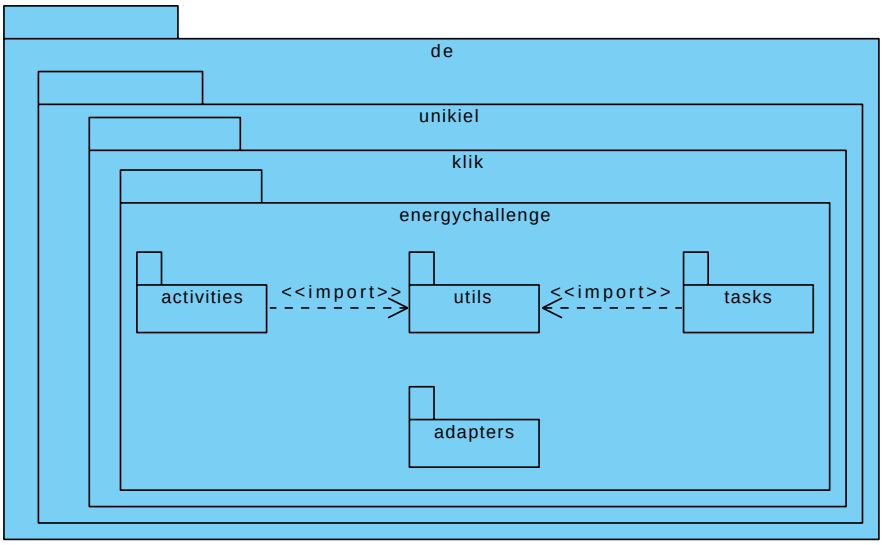

Abbildung 4: Pakete der Android App

3 Ordner Struktur

4 Klassen

Informationen Generell sind die Klassen aufgeteilt in unterschiedliche Bereiche. Zunächst die grobe Unterteilung in Klassen, die in Bezug zur Webseite stehen und Klassen mit Bezug zur Android App, dann die Klassen der Webseite weiter verfeinert in

- Modellklassen: Klassen, die sich auf die Persistenz beziehen, wie beispielsweise die verschiedenen Objekte, die in der Datenbank gespeichert werden.
- Controllerklassen: Klassen, die dazu dienen Nutzeranfragen und Darstellungen zu verwalten, jedoch auf die Datenbank nur lesend zugreifen. Diese Klassen sind zudem dafür verantwortlich die Anzeige (mittels index() Methode) der jeweiligen Daten, auf die sie sich beziehen, zu steuern. (Dieser Umstand trifft auf alle Controllerklassen zu und wird daher in der Beschreibung der einzelnen Klassen sowie bei den Methoden nicht jedes Mal explizit erwähnt.)
- Serviceklassen: Klassen, die Methoden enthalten, die die Datenpersistenz beeinflussen, also mit speicherndem und änderndem Zugriff.

4.1 Webseite

4.1.1 Modellklassen

Activity Enthält alle Informationen über eine Energiesparaktivität, die von Benutzern erledigt werden kann.

- Attribute:
  - description: Ein String, der die Aktivität beschreibt.
  - points: Eine natürliche Zahl von 1 bis 5, die angibt wie viele Punkte die Erledigung der Aktivität wert ist.
  - duration: Eine Zeitangabe, die angibt, wie lange die Aktivität nach Ausführung gesperrt ist, bevor sie wieder erledigt werden kann.

CompletedActivity Beinhaltet Informationen über von Benutzern abgeschlossene Aktivitäten.

- Attribute:
  - date: Ein DateTime Objekt, das angibt, wann die Aktivität erledigt wurde.

Profile Verwaltet Informationen bezüglich der Team- und Benutzerprofile.

- Attribute:
  - avatar: Das Profilbild, gespeichert als Byte Array.
  - name: Der Name des Benutzers oder Teams des jeweiligen Profils.
  - avatarType: Das Datenformat des Profilbildes, als String gespeichert.
  - blocked: Ein boolescher Wert, der angibt, ob das Profil gegenwärtig gesperrt ist.

User Erweitert die Klasse Profile um die persönlichen Informationen eines Benutzers und dient Apache Shiro zur Authentifikation.

- Attribute:
  - email: Die E-Mail Adresse eines Benutzers, gespeichert als String.
  - passwordHash: Die als String gespeicherte Hash-Repräsentation des Passworts eines Benutzers.
  - title: Der akademische Titel eines Benutzers als String.
  - emailNotification: Ein boolescher Wert, der angibt, ob der Benutzer E-Mail Benachrichtigungen erhälten möchte oder nicht.

Team Erweitert die Klasse Profile und beinhaltet die Informationen eines Teams.

Role Enthält Informationen über die verschiedenen Rollen auf der Webseite, die den Zugriff auf bestimmte Inhalte einschränken.

Comment Enthält Informationen zu Kommentaren bezüglich Energiesparvorschlägen.

- Attribute:
  - text: Ein String, der den Text des Kommentars enthält.
  - rating: Eine natürliche Zahl zwischen 1 und 5, die eine Bewertung des kommentierten Vorschlags darstellt.

Proposal Beinhaltet Informationen zu einem durch einen Benutzer eingereichten Energiesparvorschlag.

- Attribute:
  - description: Ein String, der den Energiesparvorschlag beschreibt.
  - points: Eine natürliche Zahl von 1 bis 5, die angibt wie viele Punkte die Erledigung der vorgeschlagenen Aktivität wert sein soll.

Institute Die Klasse, die Informationen über die Institutszugehörigkeit eines Benutzers beinhaltet.

- Attribute:
  - name: Der Name des Instituts als String.

Message Die Klasse, die Informationen über Nachrichten speichert, die an Benutzer gesendet werden können.

TeamInvite Erweitert Message und enthält Informationen bezüglich einer Teameinladung für einen Benutzer.

ActivityNotification Erweitert die Klasse Message und enthält Informationen über eine allgemeine Erinnerungsnachricht an einen Benutzer.

SpecificActivityNotification Erweitert Message und enthält Informationen über eine Erinnerungsnachricht an einen Benutzer, eine spezielle Aktivität betreffend.

4.1.2 Controllerklassen

AuthController Verwaltet das An- und Abmelden sowie das Registrieren von Benutzern auf der Webseite.

- Methoden:
  - login(): Diese Funktion steuert die Anmeldung von Benutzern.
  - logout(): Mit dieser Funktion können sich Benutzer von der Anwendung wieder abmelden und so ihre aktuelle Sitzung beenden.
  - register(): Diese Funktion dient dazu, Benutzer im System zu registrieren.

UserController Die Klasse, die die Einstellungen bezüglich eines Benutzerprofils verwaltet.

- Methoden:
  - edit(): Diese Methode ermöglicht dem Benutzer die Bearbeitung seiner persönlichen Einstellungen.
  - uploadAvatar(): Die Methode, die ein Benutzer verwendet, um ein neues Profilbild hochzuladen.
  - avatarImage():
  - show():
  - joinTeam(): Methode, die aufgerufen wird, wenn ein Benutzer eine Teameinladung annimmt oder einem bestehenden Team beitritt.
  - newTeam(): Eine Methode, die es einem Benutzer ermöglicht ein neues Team zu erstellen.
  - changePassword(): Die Methode, die aufgerufen wird, wenn ein Benutzer sein Passwort ändern möchte.
  - changeEmailNotification(): Die Methode, die die E-Mail Benachrichtigungseinstellungen eines Benutzers ändert.

TeamController Verwaltet die Einstellungen bezüglich eines Teamprofils.

- Methoden:
  - edit(): Diese Methode ermöglicht dem Benutzer die Bearbeitung der Einstellungen seines Teams.
  - uploadAvatar(): Die Funktion, die ein Benutzer aufruft, um ein neues Profilbild für sein Team hochzuladen.

AdminController Die Klasse, die sämtliche adminspezifische Anfragen ermöglicht und verwaltet.

- Methoden:
  - listUsers(): Methode, die eine Liste aller Benutzer aus der Datenbank holt und diese anzeigt.
  - listTeams(): Methode, die eine Liste aller Teams aus der Datenbank holt und diese anzeigt.
  - listProposals(): Methode, die eine Liste aller Energiesparvorschläge aus der Datenbank holt und diese anzeigt.
  - listActivities(): Methode, die eine Liste aller Aktivitäten aus der Datenbank holt und diese anzeigt.
  - editActivity(): Die Funktion, die dem Admin ermöglicht, eine Aktivität zu bearbeiten.
  - editProposal(): Die Funktion, die dem Admin ermöglicht, einen Energiesparvorschlag zu bearbeiten.
  - editTeam(): Die Funktion, die dem Admin ermöglicht, ein Team zu bearbeiten.
  - editUser(): Die Funktion, die dem Admin ermöglicht, einen Benutzer zu bearbeiten.
  - blockUser(): Die Methode, mit der ein Benutzer gesperrt wird.
  - blockTeam(): Die Methode, mit der ein Team (und damit automatisch alle Benutzer, die Mitglieder des Teams sind) gesperrt werden.
  - unblockUser(): Die Methode, mit der ein Benutzer reaktiviert wird.
  - unblockTeam(): Die Methode, mit der ein Team (und damit alle Mitglieder) reaktiviert werden.
  - deleteUser(): Methode, mit der der Admin einen Benutzer löscht.
  - deleteTeam(): Methode, mit der der Admin ein Team löscht.

AppController Die Klasse, die für die Kommunikation zwischen Server und App verantwortlich ist und Anfragen der App verarbeitet.

- Methoden:
  - getRankingTeams(): Methode, die die aktuelle Team-Rangliste aus der Datenbank liest und als JSON Objekt an die App sendet.

LandingController Steuert die Benutzung der Landing Page (Hauptseite).

RankingController Die Klasse, die die Benutzung der Ranglistenseite steuert.

ActivityController Verwaltet die Benutzung der Aktivitätenseite und ermöglicht beispielsweise das erledigen von Aktivitäten.

- Methoden:
  - completeActivity(): Die Methode, die eine Aktivität für einen Benutzer als erledigt markiert und dafür sorgt, dass ihm die entsprechenden Punkte gutgeschrieben werden.

ProposalController Die Klasse, die alle Aktionen rund um Energiesparvorschläge und die entsprechende Webseite verwaltet.

- Methoden:
  - new(): Die Methode, die einen neuen Energiesparvorschlag anlegt.
  - comment(): Mit dieser Funktion kann ein Benutzer einen neuen Kommentar zu einem Energiesparvorschlag abgeben.
  - list(): Zeigt eine Übersicht über alle Energiesparvorschläge.
  - view():

ProfileController Verwaltet die Profilseite eines Benutzers oder Teams.

- Methoden:
  - viewPage():

StatisticsController Verwaltet die Auswahl und Erstellung von Statistiken.

- Methoden:
  - exportToCsv(): Startet einen Download der Statistiken als .csv Datei.

4.1.3 Serviceklassen

ProposalService Die Klasse, die neue Energiesparvorschläge und Kommentare dazu ermöglicht.

- Methoden:
  - addComment(commentText : String, rating : int, author : Subject, proposalId : long): Methode, die dem durch die proposalId gegebenen Energiesparvorschlag einen Kommentar hinzufügt.
  - addProposal(description : String, points : int, author : Subject): Speichert einen neuen Energiesparvorschlag in der Datenbank.

ActivityService Ermöglicht verschiedene Aktionen zu Aktivitäten auszuführen.

- Methoden:
  - completeActivity(activityId : long, subject : Subject): Methode, die die durch die activityId gegebene Aktivität als erledigt markiert und die entsprechenden Punkte auf die Punktzahl des durch das Subjekt gegebenen Benutzers in der Datenbank addiert.

- addToFavorites(activityId : long, subject : Subject): Die Funktion, die eine Aktivität in den Favoriten eines Benutzers speichert.
- removeFromFavorites(activityId : long, subject : Subject): Die Methode, die eine Aktivität aus den gespeicherten Favoriten eines Benutzers löscht.

MessageService Die Klasse, die das Versenden von Teameinladungen und Erinnerungsnachrichten ausführt.

- Methoden:
  - inviteUserToTeam(receiverId : long, subject : Subject): Die Methode, die einem anderen Benutzer eine Teameinladung sendet.
  - remindUser(receiverId : long): Die Methode, die einem Benutzer eine generelle Erinnerungsnachricht sendet.
  - remindUser(receiverId : long, activityId : long): Methode, die einem Benutzer eine Erinnerungsnachricht bezüglich einer bestimmten Aktivität sendet.

SettingsService Die Klasse, die alle Daten- und Einstellungsänderungen eines Benutzers ausführt.

- Methoden:
  - setName(title : String, firstName : String, lastName : String, subject : Subject): Speichert den Namen eines Benutzers in der Datenbank.
  - setPassword(password1 : String, password2 : String, subject : Subject): Speichert das Passwort eines Benutzers als Hash in der Datenbank.
  - setAvatar(avatar : def, subject : Subject): Speichert das Profilbild eines Benutzers als Byte Array in der Datenbank.
  - setInstitute(instituteID : long, subject : Subject): Speichert das Institut eines Benutzers in der Datenbank.
  - setTeam(teamId : long, subject : Subject): Speichert das Team eines Benutzers in der Datenbank.
  - setEmailNotification(emailNotification : boolean, subject : Subject): Speichert die Einstellungen bezüglich E-Mail Benachrichtigung eines Benutzers in der Datenbank.
  - createTeamAndJoin(name : String, avatar : def, subject : Subject): Speichert ein neues Team in der Datenbank und das Team zudem als Team des Erstellers.

TeamService Ermöglicht die Änderung des Avatars und Namens eines Teams.

- Methoden:
  - setName(naem : String, teamId : long): Speichert den Namen eines Teams in der Datenbank.
  - setAvatar(avatar : def, subject : Subject): Speichert das Profilbild eines Teams als Byte Array in der Datenbank.

AdminService Führt alle Administrator-spezifischen Operationen aus, die die Datenpersistenz verändern.

- Methoden:
  - createActivity(description : String, points : int, duration : Duration): Speichert eine neue Aktivität in der Datenbank.
  - createActivityFromProposal(description : String, points : int, duration : Duration, proposalId : long): Speichert eine neue Aktivität in der Datenbank, löscht anschließend den Energiesparvorschlag, dem diese entstammt und schreibt dem Autor des Vorschlags 2 Punkte gut.
  - deleteActivity(activityId : long): Löscht eine Aktivität aus der Datenbank.
  - editActivity(activityId : long, description : String, points : int, duration : Duration): Verändert eine bestehende Aktivität in der Datenbank.
  - deleteProposal(proposalId : long): Löscht einen Energiesparvorschlag aus der Datenbank.
  - blockUser(userId : long): Setzt den blocked Wert eines Benutzers auf wahr.
  - unblockUser(userId : long): Setzt den blocked Wert eines Benutzers auf falsch.
  - deleteUser(userId : long): Löscht einen Benutzer aus der Datenbank.
  - blockTeam(teamId : long): Setzt den blocked Wert eines Teams auf wahr.
  - unblockTeam(teamId : long): Setzt den blocked Wert eines Teams auf falsch.
  - deleteTeam(teamId : long): Löscht ein Team aus der Datenbank.

PageViewService Speichert die Seitenaufrufe der einzelnen URLs in der Datenbank.

- Methoden:
  - viewPage(url : String): Speichert einen Seitenaufruf der gegebenen URL in der Datenbank.

4.2 Android App

Die Klassen in *Activities*, *Adapters* sowie die Klasse *AccessServerTask* erweitern vordefinierte Android-Klassen. Hierdurch werden viele Methoden definiert, die die vordefinierten Methoden überschreiben, wie zum Beispiel *onCreate* oder *onDestroyView*. Diese werden im Klassendiagramm und der folgenden Beschreibung nicht explizit aufgeführt.

4.2.1 Activities

MainActivity Die *Activity*, die für die Darstellung der Hauptfunktionen zuständig ist. Das bedeutet in ihr wird die Navigation aufgerufen und die meisten Funktionen, die als *Fragment* dargestellt werden.

LoginActivity Die *Activity*, über die die Anmeldung erfolgt.

SearchActivity Dient zur Ausführung einer Suche und dem Anzeigen der Ergebnisse.

ProposalActivity Dient zur Anzeige eines Energiesparvorschlags inklusive seiner Bewertungen und Kommentare. Ermöglicht außerdem das Bewerten und Kommentieren von Energiesparvorschlägen.

TeamProfilActivity Zeigt das Profil eines Benutzers an.

UserProfilActivity Zeigt das Profil eines Teams an.

MainFragment Darstellung des Haupt-*Fragments*.

MyProfilFragment Zeigt das eigene Profil an.

RankinglistFragment Zeigt das *Fragment* für die Benutzer- bzw. Teamrangliste an.

TeamRankingsTabFragment Zeigt die aktuelle Team-Rangliste an.

UserRankingsTabFragment Zeigt die aktuelle Benutzer-Rangliste an.

ProposalFragment Zeigt alle Energiesparvorschläge an.

NavigationFragment Die Navigation innerhalb der Haupt-*Activity*.

4.2.2 Adapters

FavoriteActivitiesAdapter Zuständig für die Anzeige der favorisierten Aktivitäten. Von hier aus sollen auch Aktivitäten ausgeführt werden können, wenn sie angeklickt werden.

ActivitiesAdapter Zuständig für die Anzeige aller Aktivitäten. Bei einem Klick auf eine Aktivität wird diese ausgeführt.

UserRankingsAdapter Zuständig für die Anzeige der Benutzer-Rangliste. Bei einem Klick auf einen Benutzer soll eine *UserProfilActivity* erstellt werden, in der der angeklickte Benutzer ausgeben wird.

TeamRankingsAdapter Zuständig für die Anzeige der Team-Rangliste. Bei einem Klick auf ein Team soll eine *TeamProfilActivity* erstellt werden, in der das angeklickte Team ausgeben wird.

ProposalsAdapter Zuständig für die Anzeige der Energiesparvorschläge. Bei einem Klick auf einen Vorschlag soll eine *ProposalActivity* erstellt werden, in der das angeklickte Team ausgeben wird.

4.2.3 Tasks

AccessServerTask Eine Klasse, die eine abstrakter Serveranfrage darstellt. Alle Serveranfragen-Klassen erben von dieser Klasse und implementieren die abstrakten Methoden *createServerRequest()* und *handleServerResponse()* in welchen eine Server-Anfrage gestellt wird und eine Server-Antwort verarbeitet wird. Die Klasse kümmert sich um den Verbindungsaufbau, die Antwort und das Parsen der Daten, so dass die abgeleiteten Methoden sich nicht mehr darum kümmern müssen.

GetTeamProfileTask Die Klasse, die dafür zuständig ist, ein Teamprofil vom Server zu laden und in der *TeamProfilActivity* anzuzeigen.

GetUserProfileTask Die Klasse, die dafür zuständig ist, ein Benutzerprofil vom Server zu laden und in der *UserProfilActivity* bzw. im *MyProfilFragment* anzuzeigen.

GetTeamRakingTask Die Klasse, die dafür zuständig ist, das Team-Ranking vom Server zu holen und an den *TeamRankingsAdapter* weiterzureichen.

GetUserRakingTask Die Klasse, die dafür zuständig ist, das Benutzer-Ranking vom Server zu holen und an den *TeamRankingsAdapter* weiterzureichen.

GetActivitiesTask Die Klasse, die dafür zuständig ist, die Liste aller Aktivitäten vom Server zu holen und an den *ActivitiesAdapter* weiterzureichen.

GetFavoriteActivitiesTask Die Klasse, die dafür zuständig ist, die Liste der favorisierten Aktivitäten vom Server zu holen und im *MainFragment* anzuzeigen.

GetProposalsTask Die Klasse, die dafür zuständig ist, die Energiesparvorschläge vom Server zu holen und an den *ProposalsAdapter* weiterzureichen.

GetProposalTask Die Klasse, die dafür zuständig ist, einen konkreten Energiesparvorschlag vom Server zu laden und in der *ProposalActivity* anzuzeigen.

SearchTask Die Klasse, die dafür zuständig ist, eine Suchanfrage an den Server zu stellen und das Ergebnis in der *SearchActivity* anzuzeigen.

DoActivityTask Die Klasse, die dafür zuständig ist, dem Server mitzuteilen, dass ein Benutzer eine Aktivität ausgeführt hat.

CommentProposalTask Die Klasse, die dafür zuständig ist, dem Server mitzuteilen, dass ein Benutzer einen Energiesparvorschlag bewertet oder kommentiert hat.

4.2.4 Utils

IoX Klasse, die Methoden für generelles Input/Output bereitstellt. Das ist zum aktuellen Zeitpunkt nur die Methode *readInputStream*, die einen *InputStream* in einen String umwandelt.

ServerRequest Klasse, die eine Serveranfrage modelliert. Eine Serveranfrage besteht aus einem Adressaten auf dem Server und aus der eigentlichen Anfrage. Die eigentliche Anfrage ist ein JSON-Objekt.

5 Dynamisches Verhalten

5.1 Login/Registrierung

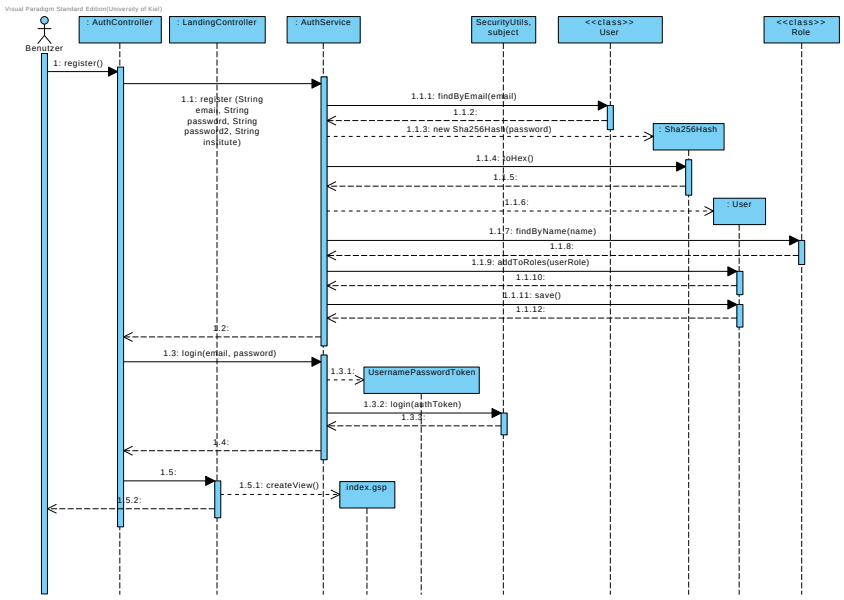

Abbildung 5: Ein Team erstellen

5.2 Aktivität erledigen

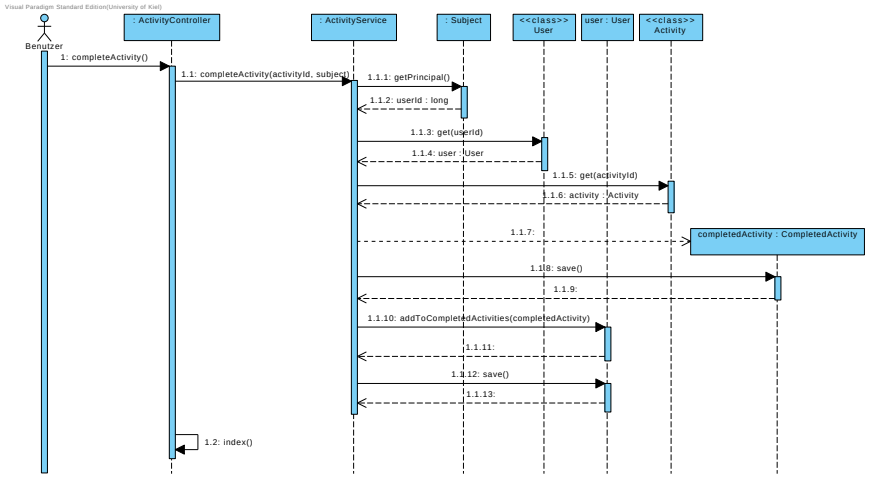

Abbildung 6: Eine Aktivität erledigen

Der eingeloggte Benutzer gelangt über einen Klick auf „Aktivitäten“ auf die Aktivitätenseite. Auf dieser findet er die Liste mit Aktivitäten. Durch das Klicken auf die zu erledigende Aktivität werden dem Benutzer die entsprechenden Punkte gutgeschrieben und die erledigte Aktivität wird für den Benutzer gesperrt bis die festgelegte Sperrphase abgelaufen ist. Der Benutzer wird dadurch außerdem wieder auf die Aktivitätenseite zurück geleitet.

Alternativ ist das Erledigen favorisierter Aktivtäten auch auf der Landingpage möglich.

5.3 Rangliste ansehen

Der eingeloggte Admin kann ein Team über einen speziellen Button, welcher die Funktion blockTeam(teamId) aufruft, sperren. Dabei wird das Team und nacheinander jedes einzelne Teammitglied gesperrt. Nachdem dies geschehen ist, tauchen auch das Team und deren Mitglieder nicht mehr in der Teilnehmerliste auf.

5.4 Team erstellen

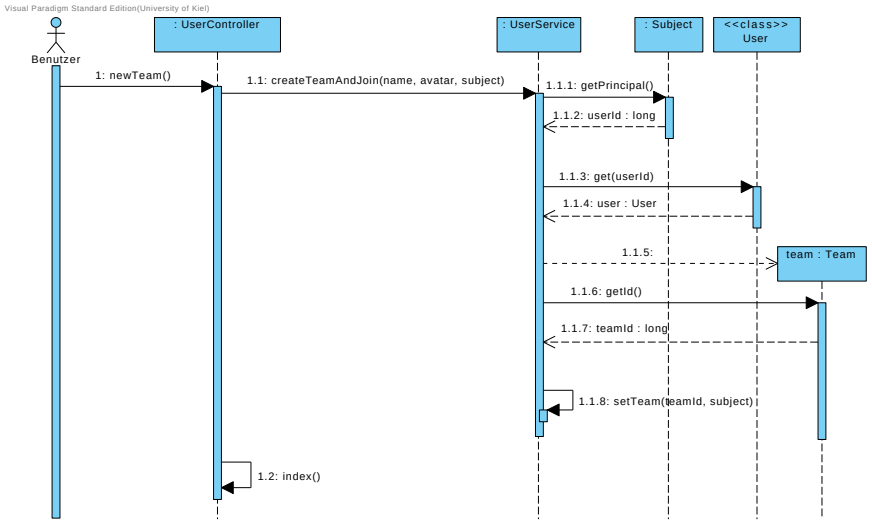

Abbildung 7: Ein Team erstellen

Der eingeloggte Benutzer, welcher noch kein Mitglied eines Teams ist, kann auf seiner Landingpage über den Link „Team erstellen“ ein neues Team generieren. Der Benutzer gelangt auf eine leere Teamprofilseite, in der er den Teamnamen, ein Teamfoto und eine Beschreibung des Teams eingeben kann. Anschließend bestätigt der Benutzer über einen Klick auf „Team gründen“ sein Team. Mit der Bestätigung ist der Benutzer automatisch dem Team beigetreten. Die Teamprofilseite aktualisiert sich und der Benutzer sieht seine neue Teamprofilseite mit ihm als Mitglied.

5.5 Team blocken

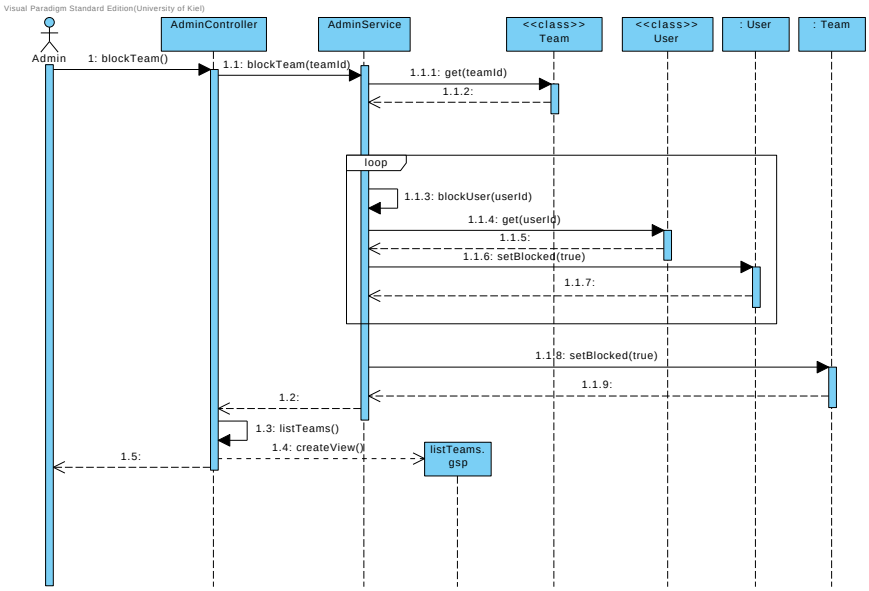

Abbildung 8: Ein Team erstellen

Der eingeloggte Admin kann ein Team über einen speziellen Button, welcher die Funktion blockTeam(teamId) aufruft, sperren. Dabei wird das Team und nacheinander jedes einzelne Teammitglied gesperrt. Nachdem dies geschehen ist, tauchen auch das Team und deren Mitglieder nicht mehr in der Teilnehmerliste auf.

5.6 Vorschlag erstellen

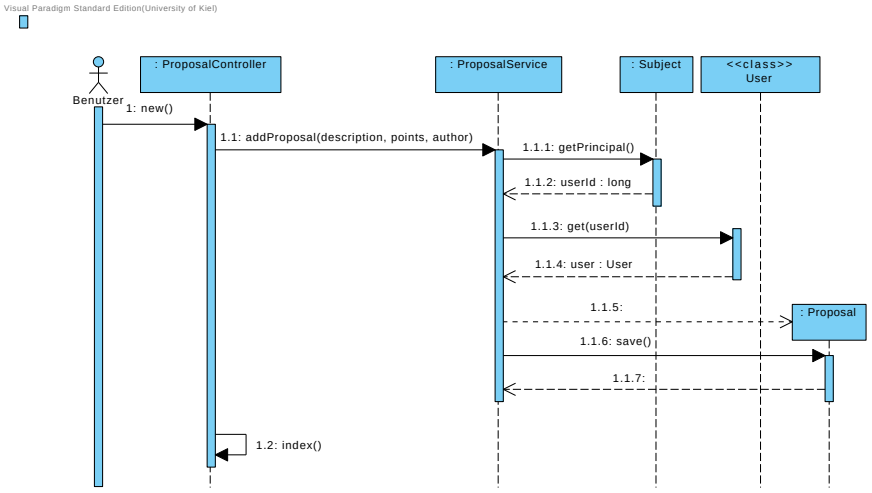

Abbildung 9: Einen Vorschlag erstellen

Der eingeloggte Benutzer, kann auf der Vorschlagsseite über den Link “eine neue Aktivität vorschlagen” einen neuen Vorschlag für eine Aktivität einreichen. Der Benutzer gelangt über den Link auf eine Seite, auf welcher er eine Beschreibung und eine Punktzahl für seinen Vorschlag eingeben kann. Auf dieser Seite kann der Benutzer durch Klick auf “Vorschlag einreichen” den Vorschlag einreichen. Anschließend sieht der Benutzer die Bestätigung “Vorschlag erfolgreich eingereicht” und wird auf die Vorschlagsseite zurückgeleitet, wo er seinen Vorschlag einesehen kann.

5.7 Vorschlag bewerten

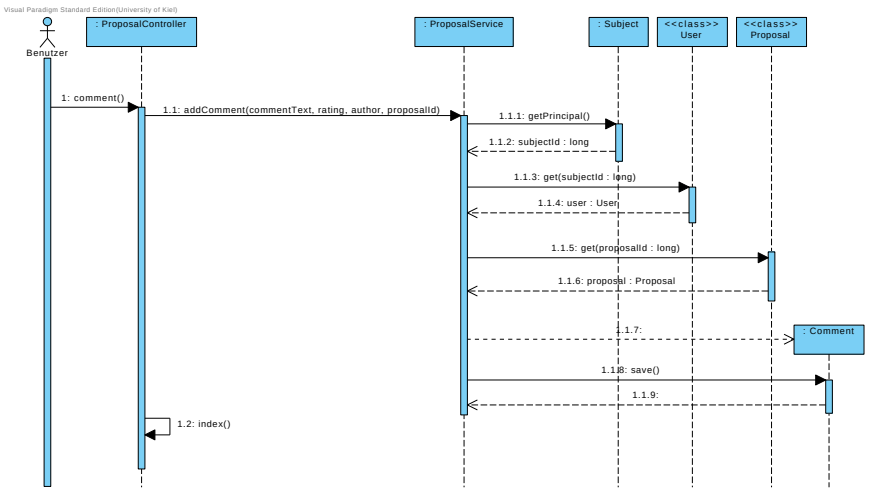

Abbildung 10: Einen Vorschlag bewerten

Der eingeloggte Benutzer, kann auf der Vorschlagsseite über den Link “diesen Vorschlag bewerten”, welcher bei jedem Vorschlag angezeigt wird, eine Bewertung zu jedem eingereichten Vorschlag abgeben. Durch Klick auf diesen Link gelangt der Benutzer auf eine Seite, auf welcher er ein Kommentar und/oder eine Bewertung in Form von 1 bis 5 Sternen abgeben kann. Auf dieser Seite kann der Benutzer durch Klick auf “Bewertung abgeben” die Bewertung abgeben. Anschließend sieht der Benutzer die Bestätigung “Bewertung abgegeben” und wird auf die Vorschlagsseite zurückgeleitet.

5.8 Statistiken exportieren

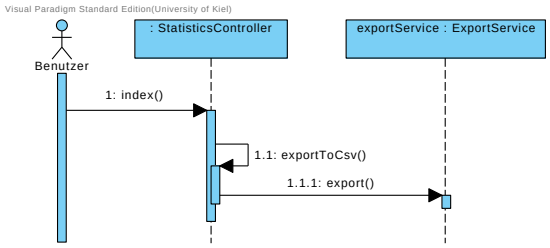

Abbildung 11: Ein Team erstellen

Der eingeloggte Benutzer klickt auf Statistik und gelangt auf die Statistikseite, wo er auf den Button „Herunterladen“ klickt. Nun wird dem Benutzer ein Download zur Verfügung gestellt, welcher ihm ermöglicht, die Statistiken in einer „.csv“-Datei auf dem eigenen Computer zu speichern.

5.9 Vorschlag in Aktivität umwandeln

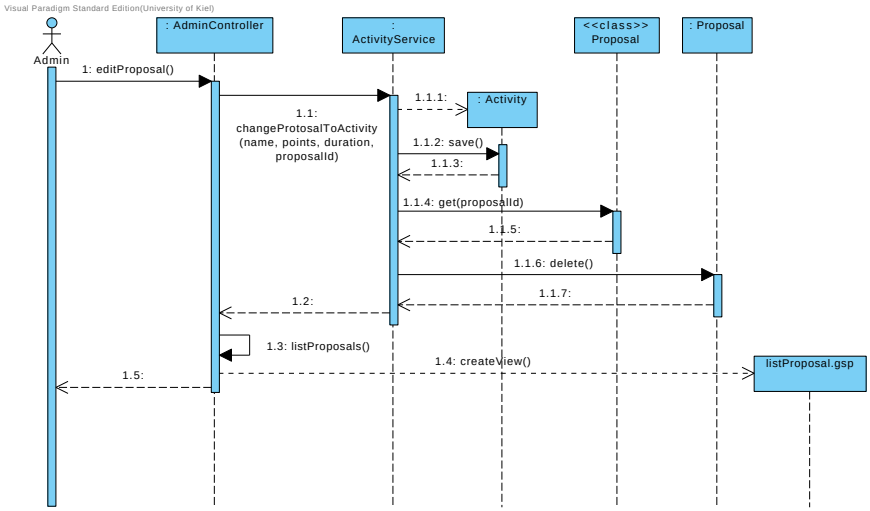

Abbildung 12: Ein Team erstellen

5.10 App: Benutzerrangliste ansehen

Die Serveranfragen in der App funktionieren alle nach dem selben Muster. Benutzerrangliste ansehen steht hier exemplarisch für weitere Anwendungsfälle dieser Art.

Wenn der Nutzer in der App zur Benutzerrangliste navigiert, wird diese automatisch bei der Erstellung des *Fragments* geladen. Dabei wird auf ein *GetUserRankingTask* die Methode *execute()* aufgerufen. *GetUserRankingTask* erbt von *AccessServerTask*, die wiederum von der Androidklassse *AsyncTask* erbt. In dem *AccessServerTask* wird die Methode *doInBackground()* aufgerufen, die wiederum die Methode *createServerRequest* aufruft, die im *GetUserRankingTask* definiert ist.
Sobald die *doInBackground()*-Methode fertig ausgeführt wurde, wird (vom *AsyncTask*) die Methode *onPostExecute()* aufgerufen, die *handleServerResponse()* des *GetUserRankingTask* ausführt. Hier wird dann die Darstellung im *Fragment* getätigt.
Der Vorteil dieser Art der Implementierung ist, dass nur einmal eine generelle Serverabfrage definiert werden muss, und sich keine weitere Gedanken um dessen Implementierung gemacht werden müssen. Deswegen ist die Durchführung der HTTP-Abfrage hier auch nicht weiter aufgeführt.

6 Anhang

6.1 Klassen Diagramme

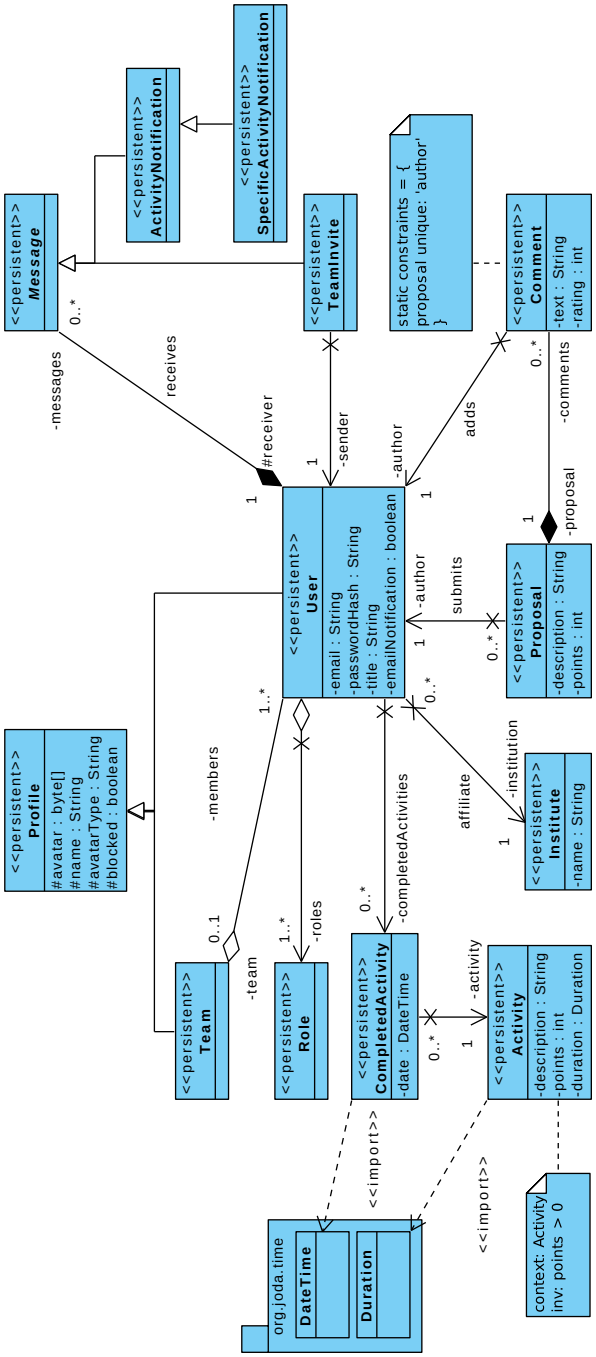

Abbildung 13: Persistenzklassen

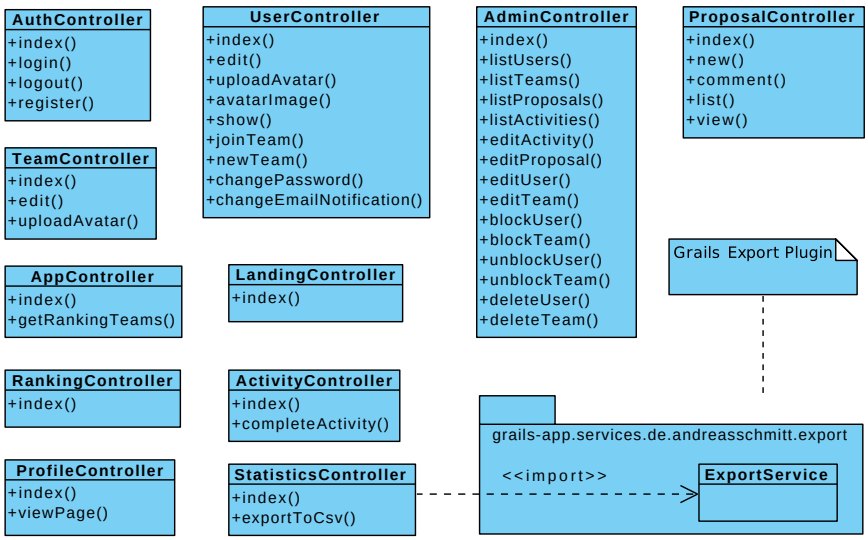

Abbildung 14: Contollerklassen

Abbildung 15: Serviceklassen