hochschule hof

University of Applied Sciences

WS 14/15 SW2

OSM

Online Sport Management

Ein Programm zum Verwalten von Sportkursen - Konstruktion

Alexander Pöhlmann
Lothar Mödl
Tim Pohrer
Burak Erol
HOCHSCHULE HOF

INHALTSVERZEICHNIS

1 FUNKTIONALITÄT

1.1 REGISTRIERUNG

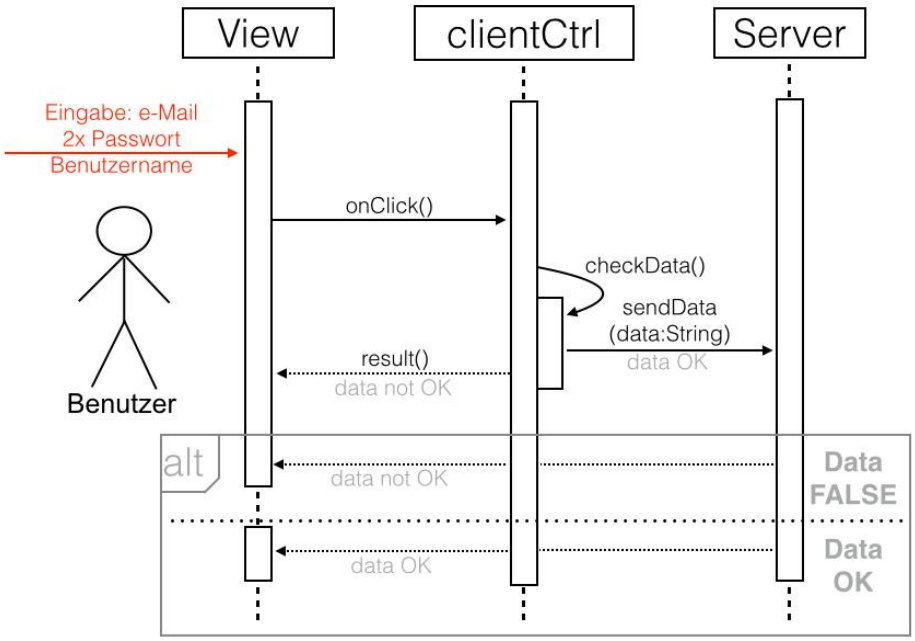

Abb. 1 Client Registrierung

Nach dem Laden der Login-Registrierung-Seite hat der Benutzer die Möglichkeit, seine Benutzerdaten einzugeben. Nach dem Betätigen eines Buttons zur Bestätigung werden die Daten aus den Textfeldern ausgelesen, und mit Hilfe der Methode „checkData()“ wird geprüft, ob:

- der Inhalt richtig ist.
- Passwort-Eingabe und -Wiederholung identisch sind.
- die E-Mail-Adresse die Endung "@hof-university.de" hat.

Bei einer gescheiterten Überprüfung der Daten werden die falschen Eingaben und das Passwort gelöscht und Hinweise dargestellt. Danach hat der Benutzer die Möglichkeit, die Eingabe zu korrigieren.

Bei erfolgreicher Überprüfung werden die Daten an den Server geschickt. Der Server überprüft die Daten (siehe Server Registrierung) ebenfalls.

Bei einer erfolgreichen Registrierung wird eine neue Seite geladen mit dem Hinweis, dass die Registrierung erfolgreich war, dass eine E-Mail an die angegebene E-Mail-Adresse geschickt wurde und dass erst der Link in der E-Mail bestätigt werden muss.

onClick() - Der „clientCtrl“ wird aufgerufen.

checkData() – Überprüft, ob Eingaben getätigt wurden und ob die eingegebenen Passwörter identisch sind. Die E-Mail-Adressen-Endung wird auf "@hof-university.de" überprüft.

sendData(data:String) - Die eingegeben Daten werden an den Server übertragen.

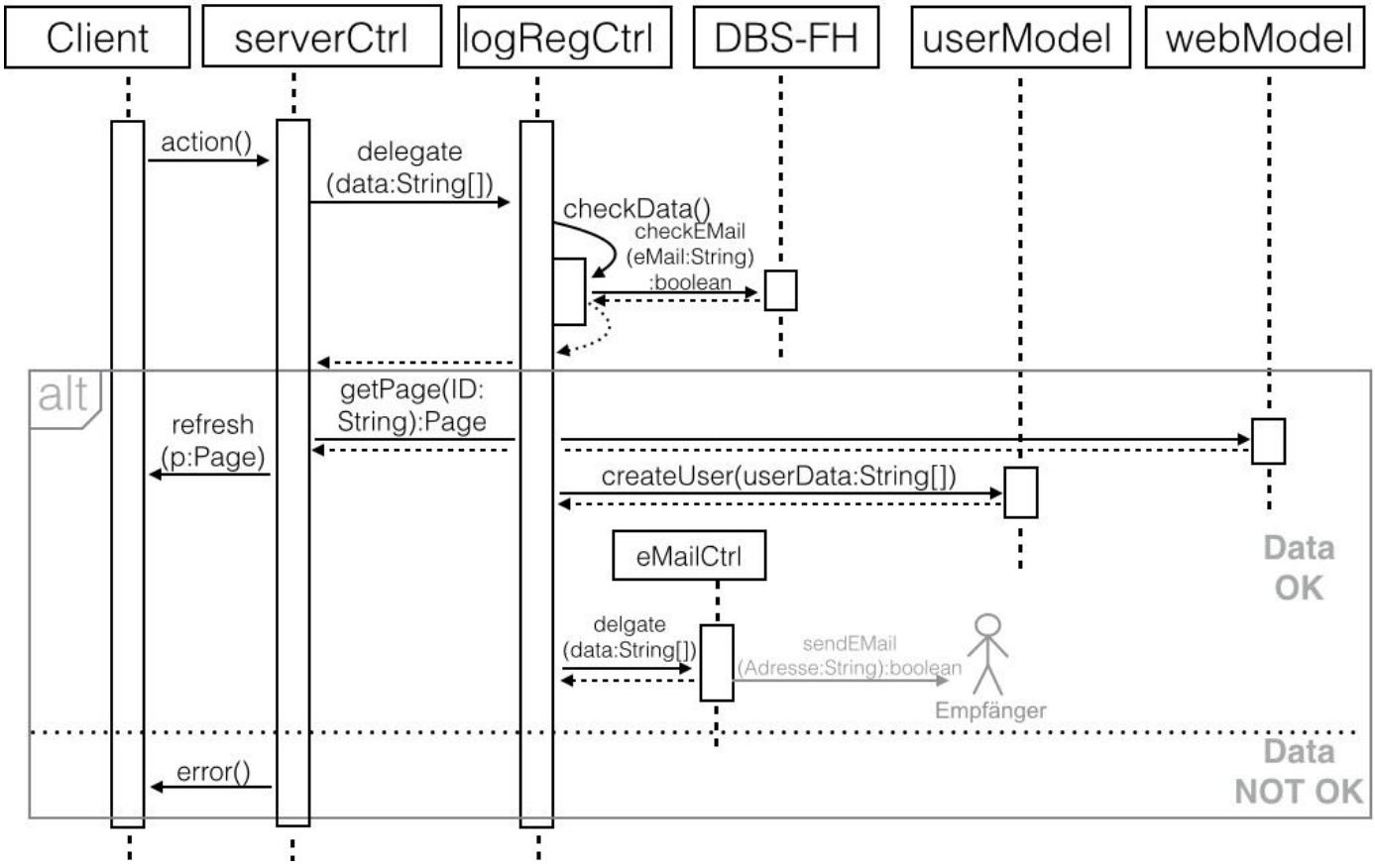

Abb. 2 Server Registrierung

Dieser Abschnitt zeigt das Vorgehen des Programmes bei der Registrierung eines Nutzers. Es ist zu beachten, dass der Ablauf der Registrierung bei Kursleiter und Teilnehmer der gleiche ist. Zu Beginn werden die Daten vom „serverCtrl“ empfangen

und an den zuständigen subCtrl "logRegCrl" weitergeleitet. Dieser überprüft, ob die Daten schon vorhanden sind oder nicht existieren. Hierbei wird nur die E-Mail-Adresse überprüft. Zum Abgleich wird die DBS-FH und das „userModel“ verwendet.

Bei einer erfolgreichen Registrierung wird eine neue Seite geladen und ein neuer Nutzer dem „userModel“ hinzugefügt. Der neu registrierte Nutzer bekommt eine Bestätigungs-E-Mail.

Bei einer fehlgeschlagenen Registrierung wird keine neue Seite geladen, sondern eine Fehlermeldung an den Client weitergeleitet.

action() - Übermittelt die Eingabe des Nutzers an den Server.
delegate(data:String[]) - Übermittelt die Daten vom „serverCtrl“ an den jeweiligen „subCtrl“.
checkData() – Überprüft, ob die E-Mail-Adresse schon einmal verwendet wurde.
checkEMail(eMail:String):boolean – Überprüft, ob die E-Mail-Adresse auch wirklich einem Kursleiter oder einem Teilnehmer gehört.
getPage(ID:String):Page - Liefert eine neue Webseite zurück.
refresh(p:Page) - Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.
createUser(userData:String[]) - Übergibt die Daten zum Erstellen des Nutzers an das „userModel“.
sendEMail(Adresse:String):boolean - Sendet eine entsprechende E-Mail an die übergebene Adresse.
error() – Gibt dem Nutzer eine Fehlermeldung zurück.

1.2 LOGIN

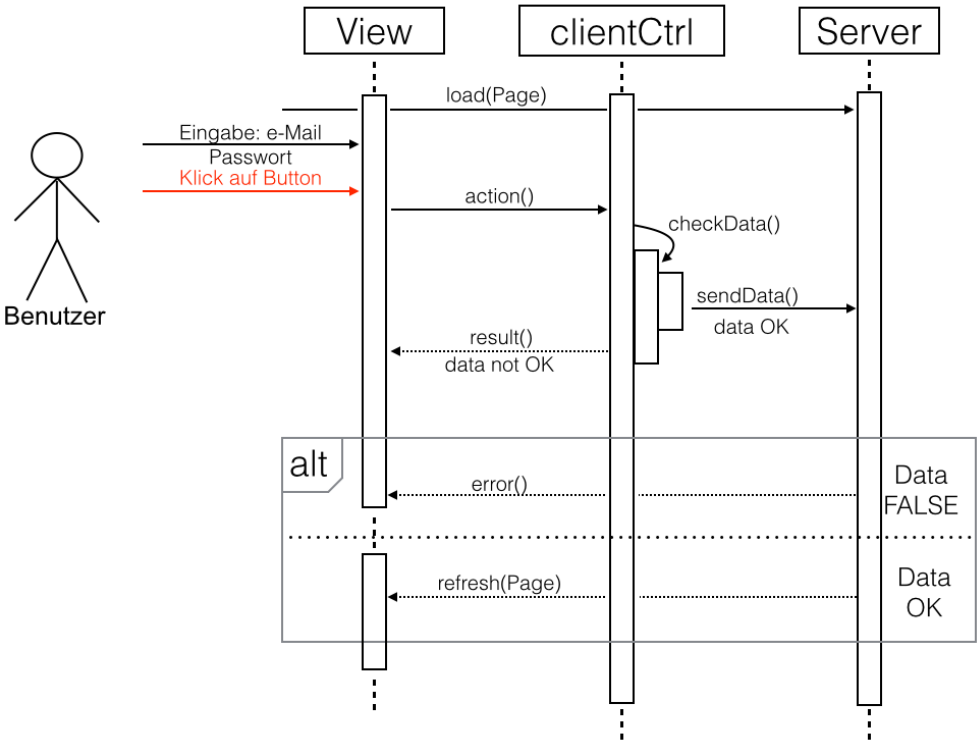

Abb. 3 Login Client

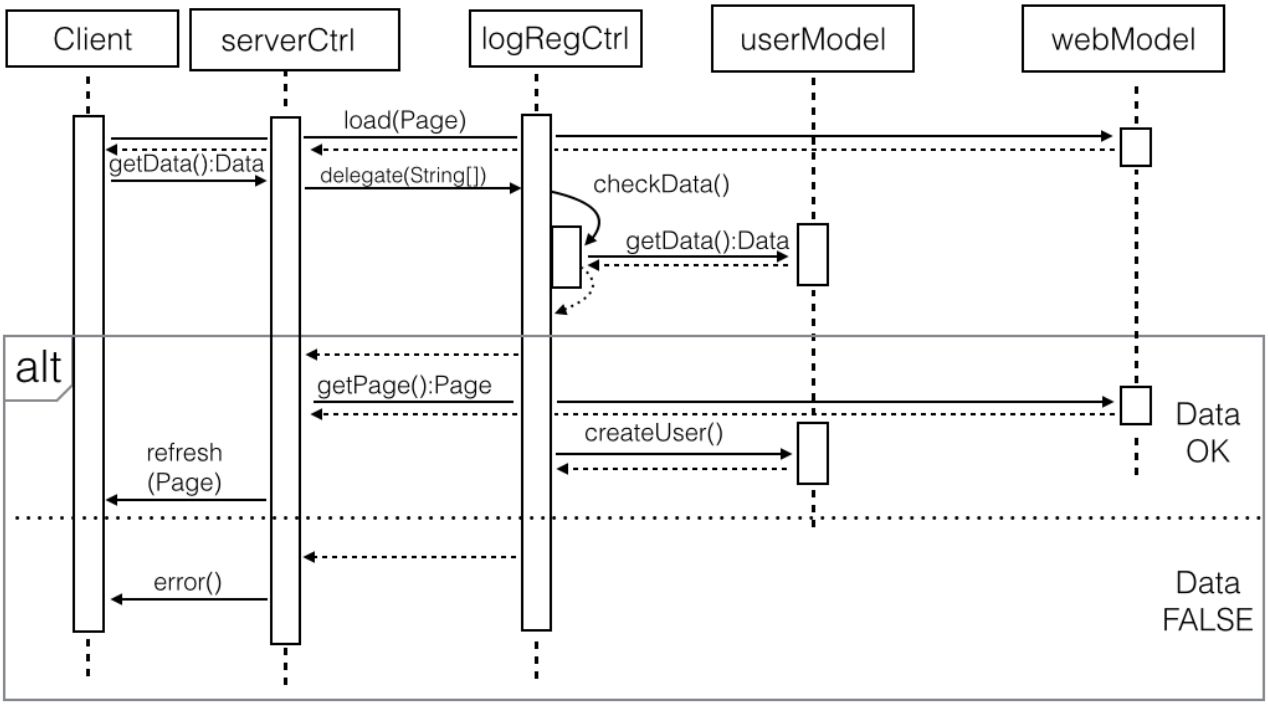

Abb. 4 Login Server

1.3 LOGOUT

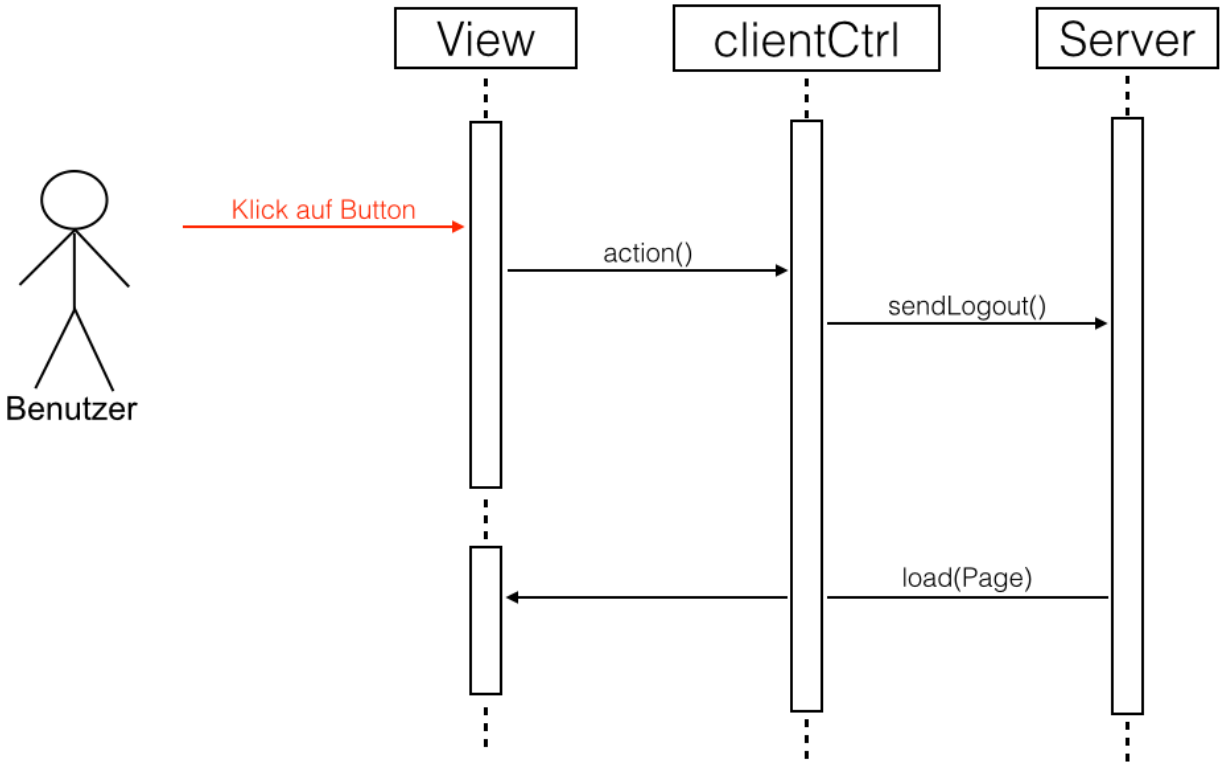

Abb. 5 Logout Client

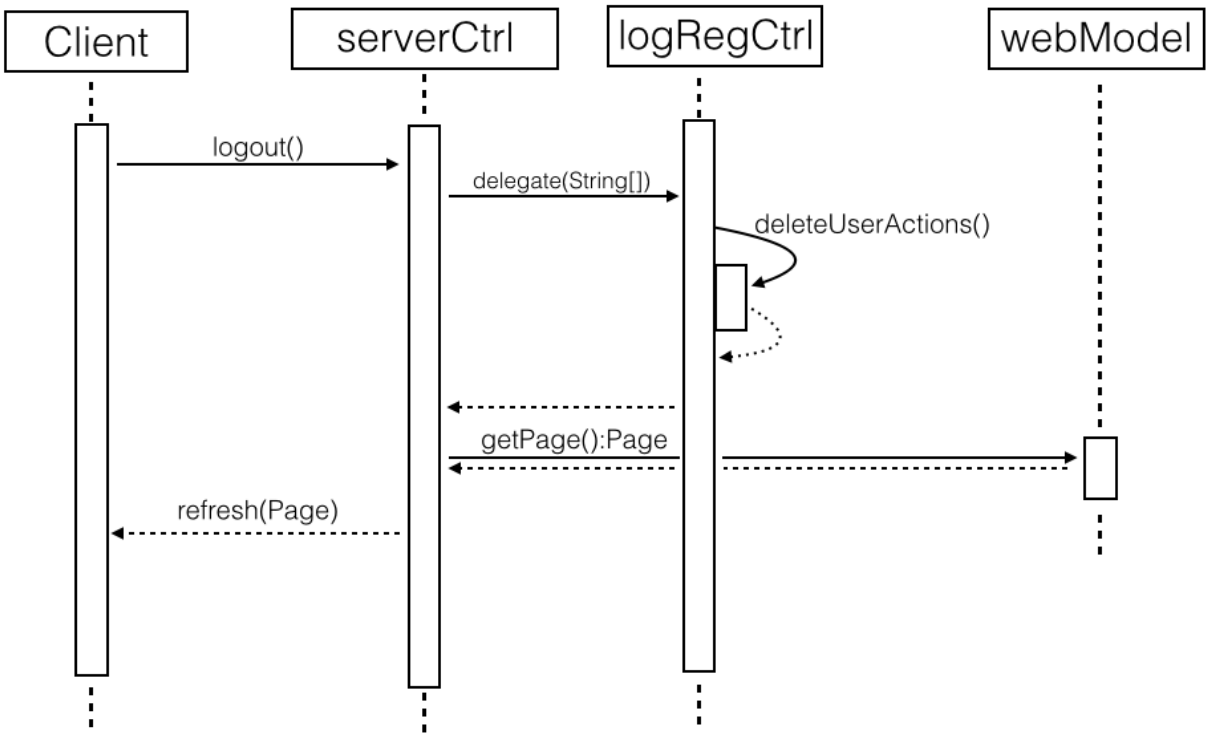

Abb. 6 Logout Server

1.4 KURS

1.4.1 KURS ANMELDEN

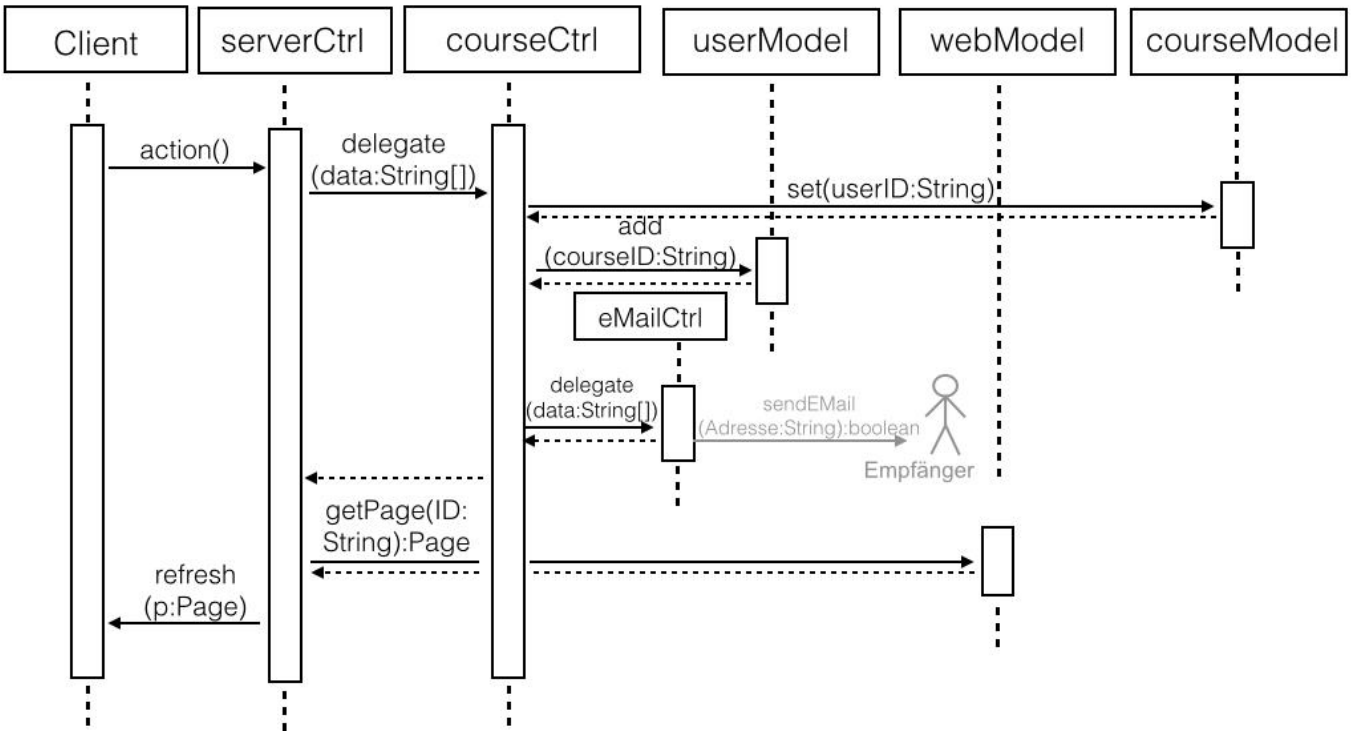

Abb. 7 Server Kurs anmelden

Wenn der Nutzer die Schaltfläche zum Einschreiben in einen Kurs betätigt, empfängt der „serverCtrl“ die Action und stößt den Kurscontroller (courseCtrl) an. Dieser fügt den Namen des Benutzers zu dem jeweiligen Kurs hinzu. Der Zähler bei dem jeweiligen Kurs wird dann automatisch erhöht. Des Weiteren wird mit der Methode "add(courseID:String)" der Kurs, zu dem sich der Benutzer anmelden will, zu dessen Kursen hinzugefügt. Am Ende dieser Aktion ändert sich die Schaltfläche, die zuvor für das Anmelden eines Kurses da war, zum Abmelden von dem jeweiligen Kurs (bezieht sich auf Spezifikation - Use-Case-Diagramm – Abb. 2).

action() - Die Auswahl der vom Nutzer gewünschten Funktion. Hier Auswahl zum Erstellen einer neuen Neuigkeit.

delegate(data:String[]) - Stößt den „subCtrl“ an und übergibt ihm alle notwendigen Daten als Argument.

set(userID:String) - Der Name des Benutzers wird bei dem entsprechenden Kurs im „courseModel“ gespeichert.

add(courseID:String) - Der Kurs wird bei dem entsprechenden Benutzer im „user-Model“ hinterlegt.

sendEMail(Adresse:String):boolean - Sendet eine entsprechende E-Mail an die übergebene Adresse.

getPage(ID:String):Page - Liefert die entsprechende Website mit den neuen Inhalten zurück.

refresh(p:Page) - Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.

1.4.2 KURS ABMELDEN

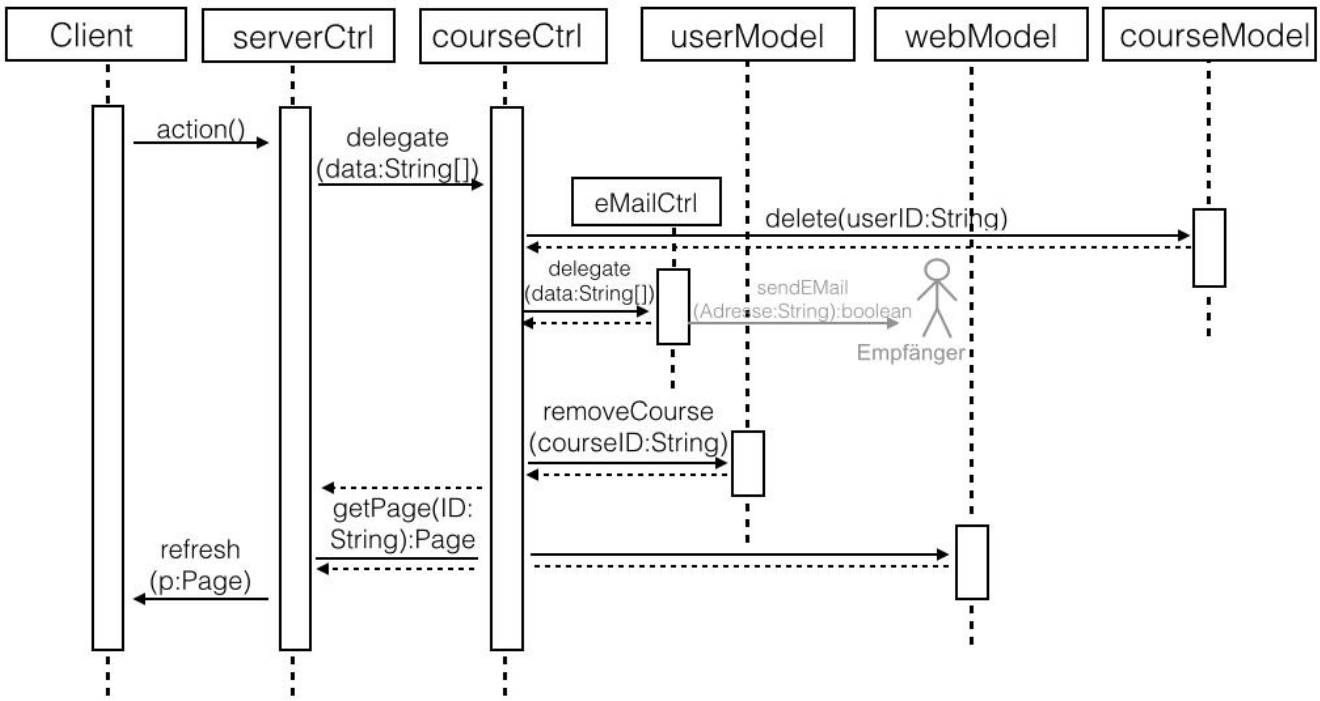

Abb. 8 Server Kurs abmelden

Jeder Benutzer hat die Möglichkeit, sich bei einem Kurs, bei dem er sich angemeldet hat, wieder abzumelden. Nach der Auswahl, aus welchem Kurs sich der Benutzer abmelden will, wird diese an den „serverCtrl“ übertragen. Dieser ruft mit Hilfe der „delegate(data:String[])“ Methode den „courseCtrl“ auf. Danach wird der Benutzer aus dem jeweiligen Kurs im "courseModel" gelöscht. Außerdem wird der Kurs beim jeweiligen Benutzer aus dem "userModel" gelöscht. Danach wird die aktualisierte Seite dem Nutzer präsentiert.

action() – Stößt den „serverCtrl“ an.

delegate(data:String[]) - Übermittelt die Daten vom „serverCtrl“ an den jeweiligen „subCtrl“.

delete(userID:String) - Der Benutzer wird aus dem jeweiligen Kurs im "courseModel" entfernt.

sendEMail(Adresse:String):boolean - Sendet eine entsprechende E-Mail an die übergebene Adresse.

romoveCourse(courseID:String) - Der Kurs, von dem sich der Nutzer abmelden will, wird aus seinem "userModel" entfernt.

getPage(ID:String):Page - Liefert die entsprechende Website mit den neuen Inhalten zurück.

refresh(p:Page) - Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.

1.4.3 KURS ANLEGEN

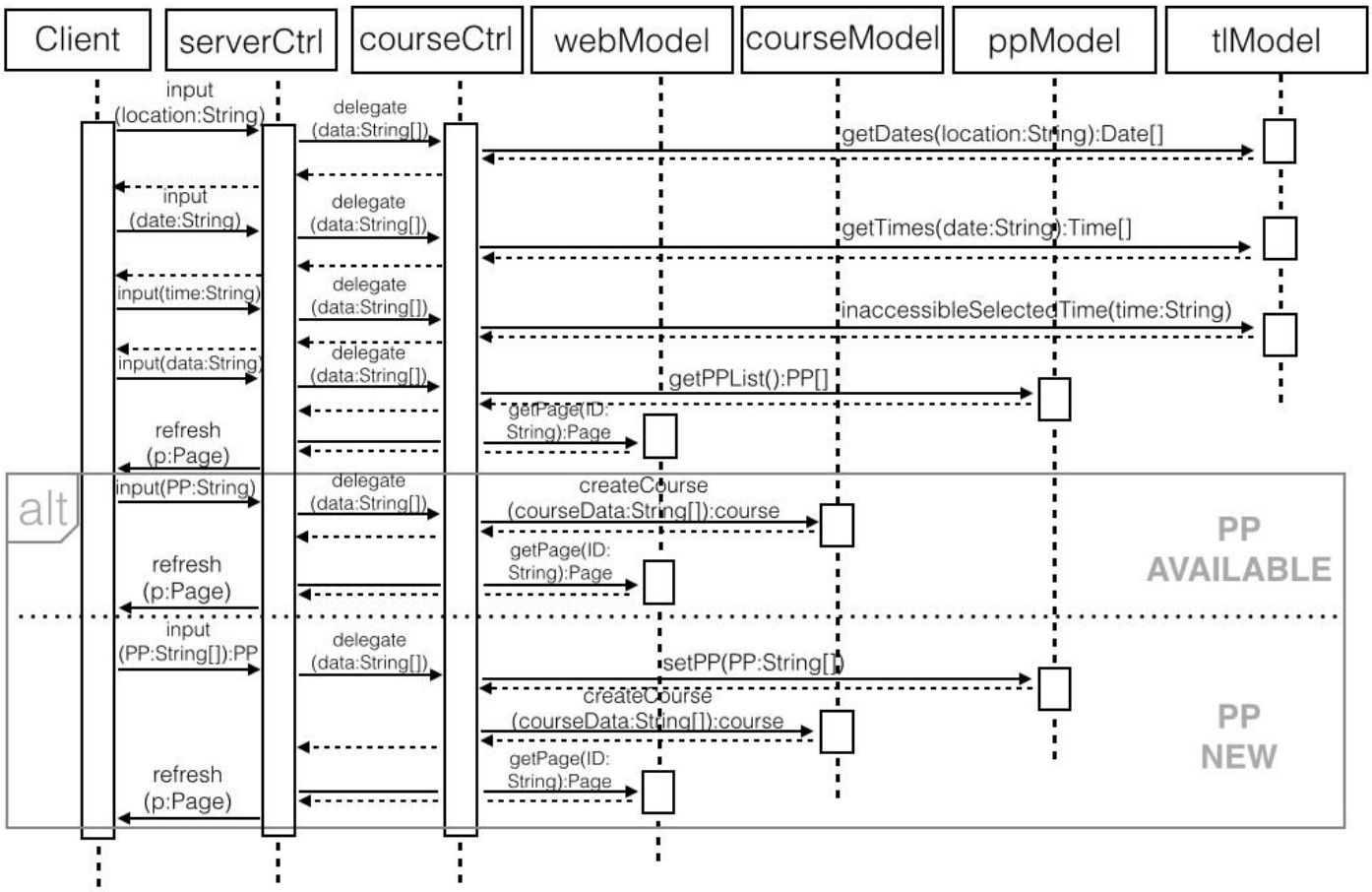

Abb. 9 Server Kurs erstellen

Begriffsklärung

ppModel - Performance Profil Model

tlModel = Time and Location Model

Jeder Sportleiter kann neue Sportkurse erstellen. Zuerst muss der Ort, an dem der Sportkurs stattfinden soll, eingegeben werden, welcher mit der Methode „input(location:String)“ an den Server übermittelt wird. Danach werden die Termine, die an dem gewählten Ort noch verfügbar sind, angezeigt. Der Sportleiter kann dann den gewünschten Wochentag auswählen. Dieser wird mit Hilfe der Methode „input(date:String) an den Server gesendet. Es werden danach die verschiedenen verfügbaren Uhrzeiten an den Ersteller des Kurses zurückgeliefert, sodass dieser wieder die gewünschte Zeit auswählen kann. Im Anschluss kann der Sportleiter noch

zusätzliche Daten eingeben, die mit Hilfe der Methode „input(data:String)“ an den Server übertragen werden.

Zu jedem neuen Sportkurs muss ein Leistungsprofil angegeben werden. Sollte es eine Sportart sein, die schon einmal angeboten wurde, d. h. es existiert schon ein Leistungsprofil, so kann der Leiter aus einer Liste das passende Profil auswählen. Sollte es eine neue Sportart sein, für die noch kein Leistungsprofil existiert, so kann ein neues erstellt werden. Es müssen dann die entsprechenden prozentualen Werte für die einzelnen Bereiche (s. Spezifikation Punkt 2.4.2) angegeben werden. Die Methode „input(PP:String[]):PP” übermittelt diese Werte dann an den Server. Wurde alles erfolgreich eingegeben, so wird der Kurs erstellt und freigegeben, damit sich die Benutzer für diesen einschreiben können. (Bezieht sich auf Spezifikation - Use-Case-Diagramm – Abb. 5)

input(String) - Übermittelt die vom Kursleiter eingegebenen Daten an den Server und empfängt sie wieder, sodass die Seite für die Eingabe nicht zerstört wird.

refresh(p:Page) - Lädt die Seite neu, die der Kursleiter zu sehen bekommt.

delegate(data:String[]) - Übermittelt die Daten vom „serverCtrl“ an den jeweiligen „subCtrl“.

getDates(location:String):Date[] - Liefert ein Array mit allen noch zur Verfügung stehenden Terminen zurück.

getTimes(date:String):Time[] - Liefert ein Array mit allen noch zur Verfügung stehenden Zeiten zurück.

inaccessibleSelectedTime(time:String) - Macht die vom Kursleiter ausgewählte Zeit im tlModel unzugänglich.

getPPList():PP[] - Liefert eine Liste aller verfügbaren Leistungsprofile zurück.

getPage(ID:String):Page - Liefert eine neue Webseite zurück.

createCourse(courseData:String[]):course - Erstellt mit dem vom Sportleiter angegebenen Daten einen neuen Kurs.

setPP(PP:String[]) - Schickt das vom Sportleiter neu erstellte Leistungsprofil in das „ppModel“.

1.4.4 KURS LÖSCHEN

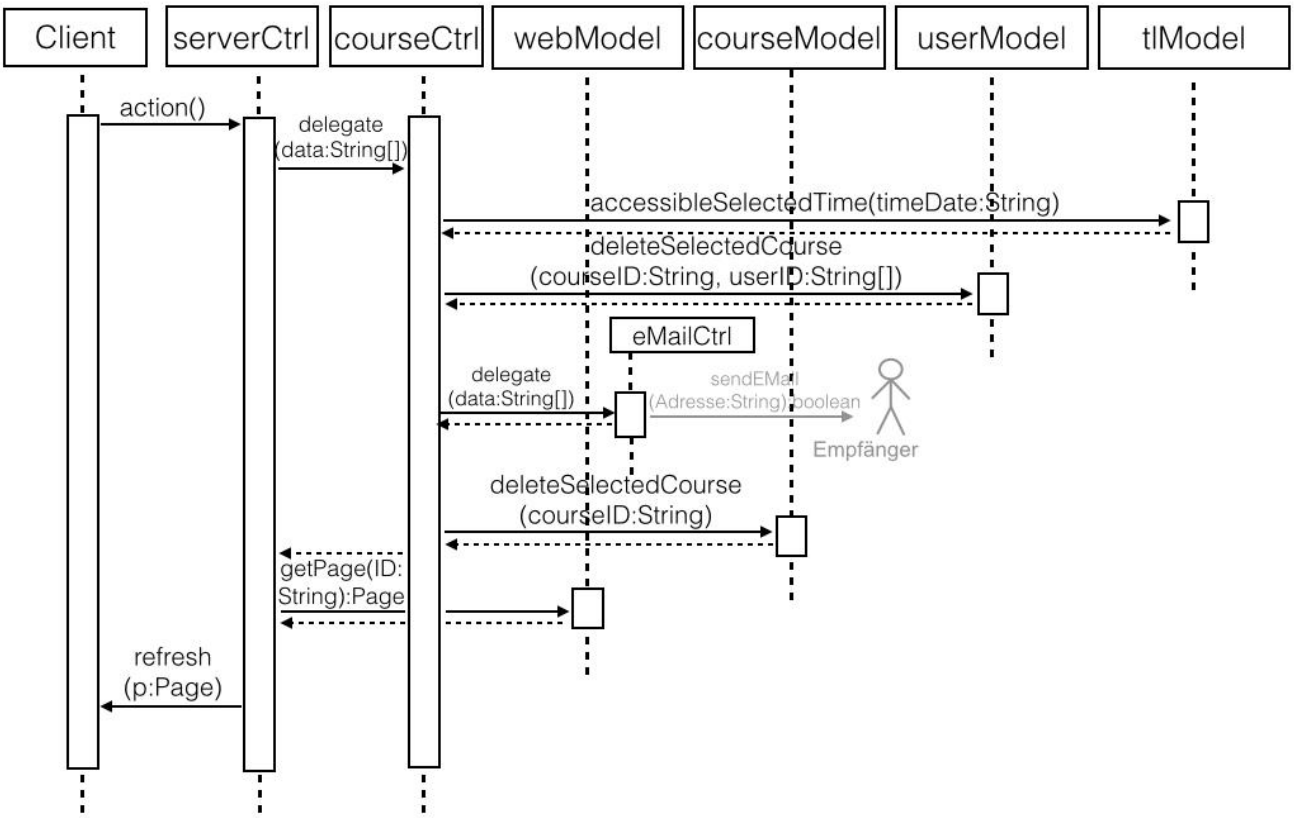

Abb. 10 Server Kurs löschen

Mit diesem Diagramm wird auf der Serverseite der Ablauf des Programmes dargestellt. Zu Beginn wird die Zeit wieder freigegeben, die der Kurs für sich beansprucht hat. Danach wird der Kurs bei allen Teilnehmern mit Hilfe der ID des Kurses und der ID der Teilnehmer entfernt. Im Anschluss bekommen alle Teilnehmer eine Benachrichtigung via E-Mail, dass dieser Kurs nicht mehr angeboten wird. Der Kurs selber wird mit der ID des Kurses als letztes gelöscht. Danach wird eine Rückmeldung an den Nutzer in Form einer neuen Seite geschickt (Bezieht sich auf Spezifikation - Use-Case-Diagramm – Abb. 4).

action() – Stößt den „serverCtrl“ an.

delegate(data:String[]) - Übermittelt die Daten vom „serverCtrl“ an den jeweiligen „subCtrl“.

accessibleSelectedTime(String) - Freigabe der Zeit zu einem entsprechenden Datum für andere Kurse. Die Uhrzeit und Datum sind im String gespeichert im Format DD:MM:YYYY||MM:SS

deleteSelectedCourse(courseID:String, userID:String[]) - Löscht bei allen Teilnehmern eines Kurses den Kurs aus der Teilnehmerliste. Mit dem String wird die „courseID“ und eine Liste der IDs der Teilnehmer gespeichert.

sendEMail(Adresse:String):boolean - Sendet eine entsprechende E-Mail an die übergebene Adresse.

deleteSelectedCourse(courseID:String) - Löscht den Kurs aus dem Speicher. In dem String ist die ID des Kurses gespeichert.

getPage(ID:String):Page - Liefert eine neue Webseite zurück.

refresh(p:Page) - Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.

1.5 PROFIL

1.5.1 PROFIL LÖSCHEN

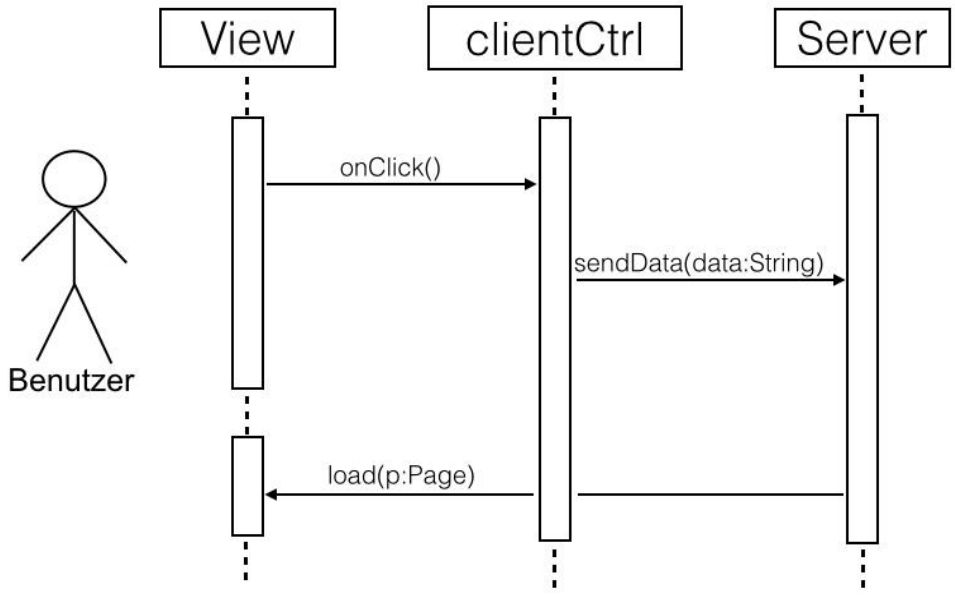

Abb. 11 Client Profil löschen

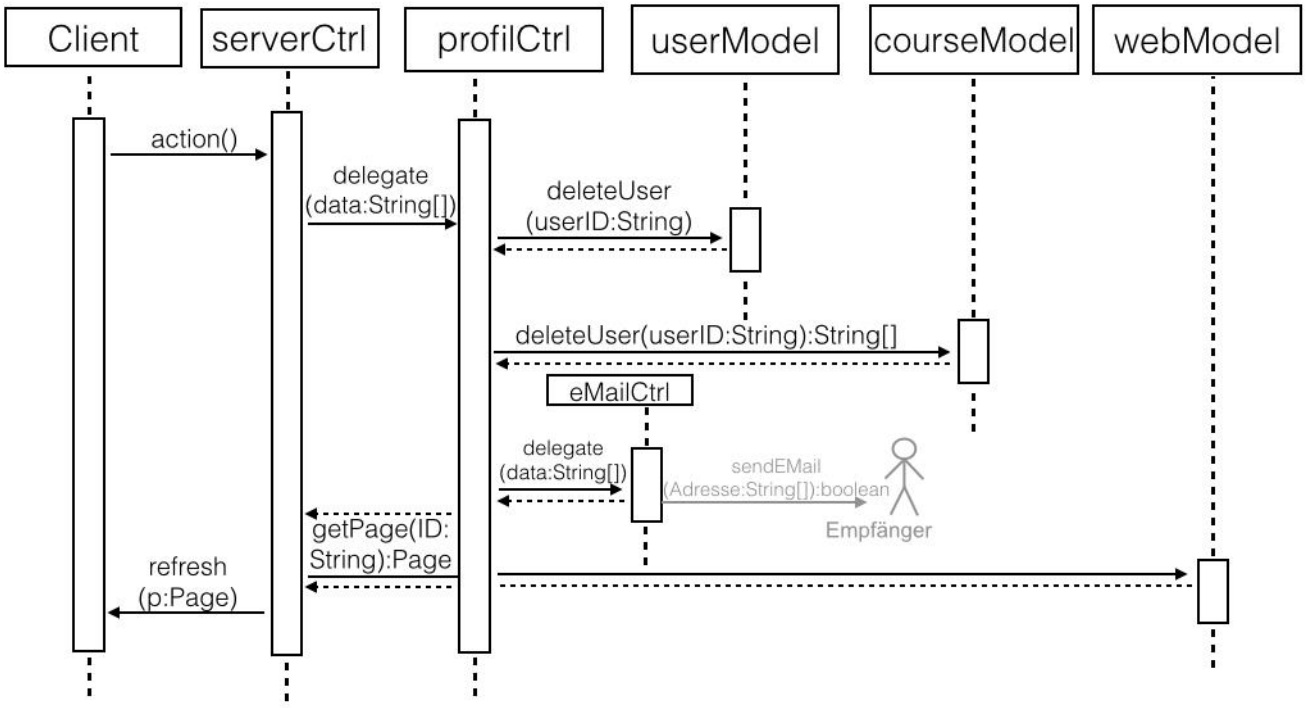

Abb. 12 Server Profil löschen

Will der Nutzer (egal ob Teilnehmer oder Leiter) sein Profil löschen, wird als Erstes sein Profil mit Hilfe seiner ID aus dem „userModel“ gelöscht („delete(userId:String)“).

Falls ein Kursleiter sein Profil löscht, werden anschließend alle Teilnehmer aus seinen Kursen entfernt („deleteUser(userId:String):String[]“), und eine Liste mit allen Teilnehmern, die aus seinen Kursen entfernt wurden, wird zurückgegeben. Danach wird eine Benachrichtigungsmail („sendEmail(Adresse:String[])):boolean“) an alle Teilnehmer, die in seinen Kursen eingeschrieben waren, und an den Leiter, der sein Profil löscht, gesendet. Anschließend wird aus dem „webModel“ eine neue Seite geladen und dem Nutzer angezeigt. Diese Seite zeigt dem Nutzer an, ob das Löschen des Profils erfolgreich war oder nicht.

Falls ein Nutzer sein Profil löscht, wird er anschließend aus allen eingeschriebenen Kursen entfernt („deleteUser(userId:String):String[]“) und bekommt eine Benachrichtigungsmail („sendEmail(Adresse:String[]):boolean“). Anschließend wird aus dem „webModel“ eine neue Seite geladen und dem Nutzer angezeigt. Diese Seite zeigt dem Nutzer an, ob das Löschen des Profils erfolgreich war oder nicht.

action() - Übermittelt die Auswahl des Nutzers an den Server.

delegate(data:String[]) - Stößt den „subCtrl“ an und übergibt ihm alle notwendigen Daten als Argument.

deleteUser(userId:String) – Löscht den Nutzer mit entsprechender ID.

deleteUser(userID:String):String[] – Entfernt den Nutzer aus allen Kursen, in denen er eingeschrieben ist.

sendEmail(Adresse:String[])):boolean – Sendet E-Mail an die Nutzer mit übergebener/n Adresse(n).

getPage(Id:String):Page - Liefert eine neue Webseite mit der entsprechender ID zurück.

refresh(p:Page) - Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.

1.6 ANDERE NUTZER EINLADEN

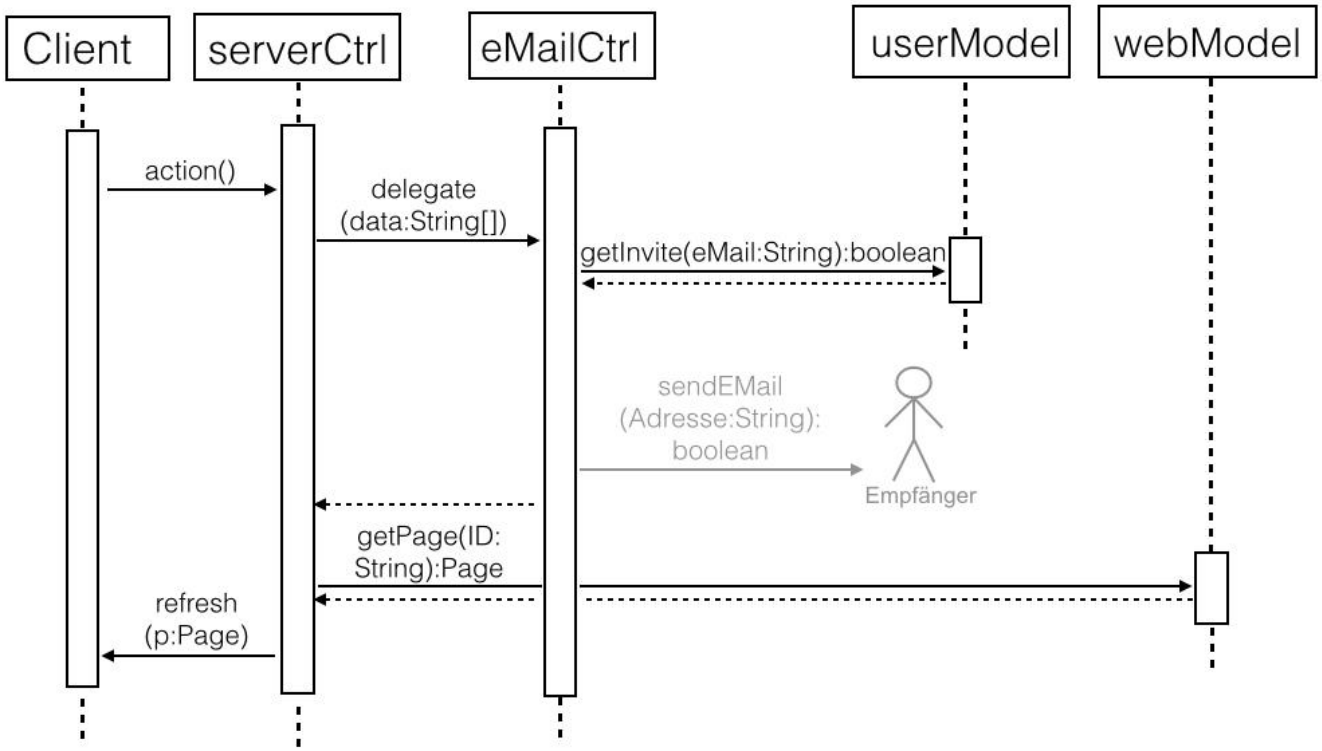

Abb. 13 Server andere Nutzer einladen

Nach dem Betätigen der Schaltfläche auf der Seite des Client und nach einer erfolgreichen Überprüfung der E-Mail-Adresse wird diese an den „serverCtrl“ geschickt. Danach werden diese und andere relevante Informationen, die die E-Mail enthalten soll, an den „subCtrl“ geschickt. Im „subCtrl“ wird überprüft, ob der Nutzer eine Nachricht empfangen möchte. Nach einer erfolgreichen Überprüfung wird die E-Mail versendet.

action() – Stößt den „serverCtrl“ an.
delegate(data:String[]) - Übermittelt die Daten vom „serverCtrl“ an den jeweiligen „subCtrl“.
getInvite(eMail:String):boolean - Überprüft, ob der Nutzer, der eingeladen werden soll, eine E-Mail bekommen möchte, und gibt dementsprechend einen Boolean-Wert zurück.
sendEMail(Adresse:String):boolean - Sendet eine entsprechende E-Mail an die übergebene Adresse.
getPage(ID:String):Page - Liefert eine neue Webseite zurück.
refresh(p:Page) - Lädt die Seite neu, die der Kursleiter zu sehen bekommt.

1.7 NEUIGKEIT

1.7.1 NEUIGKEIT LÖSCHEN

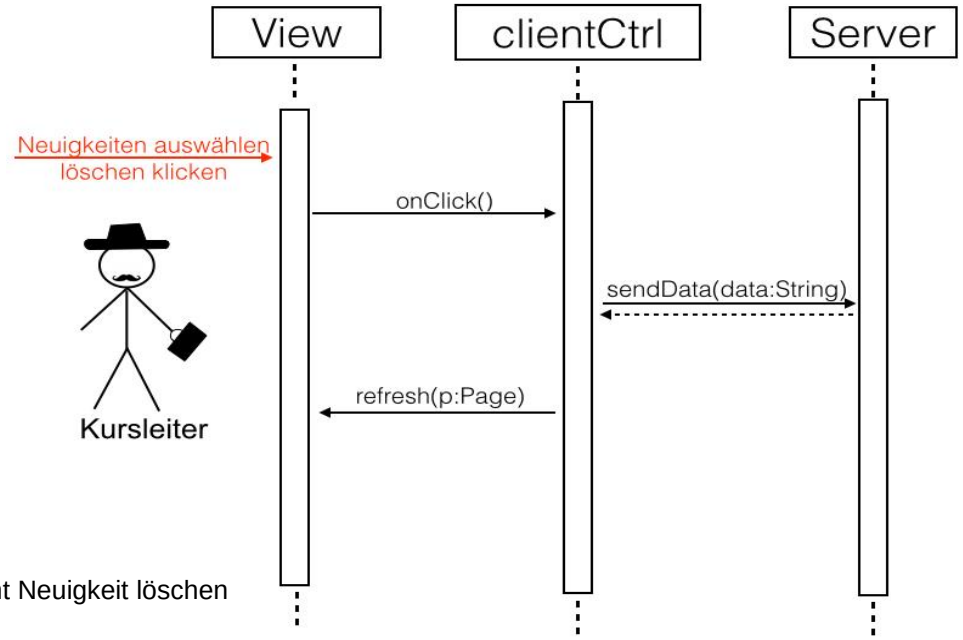

Abb. 14 Client Neuigkeit löschen

Client Neuigkeit löschen

Der Sportleiter hat die Möglichkeit, seine Neuigkeiten wieder zu löschen. Bei der Übersicht aller Neuigkeiten kann eine Neuigkeit ausgewählt und dann die Schaltfläche zum Löschen betätigt werden. Ist dies der Fall, so wird die Methode „sendData(data:String)“ aufgerufen, welche die entsprechende ID der Neuigkeit als Eingangsparameter enthält. Nach dem Löschen ist die Neuigkeit nicht wieder herzustellen.

onClick() - Der „clientCtrl“ wird aufgerufen.
sendData(data:String) - Die eingegebenen Daten werden an den Server übertragen.
refresh(p:Page) - Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.

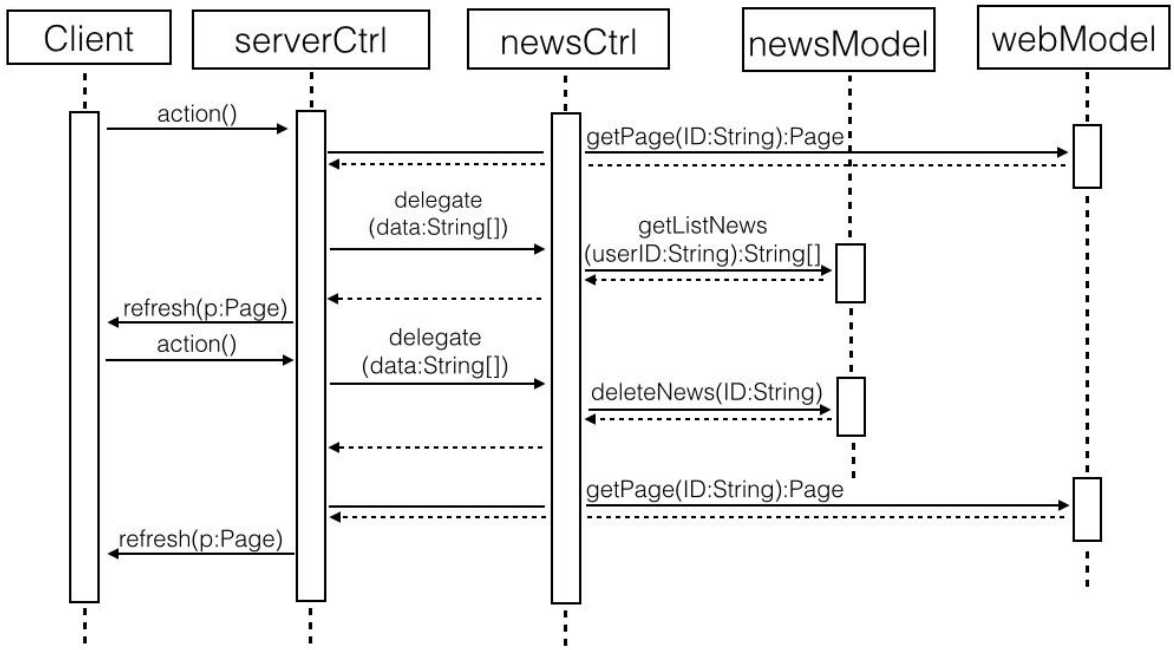

Abb. 15 Server Neuigkeit löschen

Will der Kursleiter eine Neuigkeit löschen, werden alle News, die er erstellt hat, aus dem „newsModel“ geladen („getListNews(userId:String):String[]“) und angezeigt. Danach kann der Nutzer die zu löschende Neuigkeit auswählen, welche anschließend aus dem „newsModel“ auch gelöscht wird („deleteNew(ID:String)“). Schließlich wird eine neue Seite geladen, die dem Nutzer den Erfolg der Aktion zeigt.

action() - Übermittelt die Auswahl des Nutzers an den Server.
getPage(Id:String):Page - Liefert eine neue Webseite mit der entsprechenden ID zurück.
delegate(data:String[]) - Stößt den „subCtrl“ an und übergibt ihm alle notwendigen Daten als Argument.
getListNews(userId:String):String[] – Liefert eine Liste mit allen News zurück, die der User mit der entsprechenden „userID“ erstellt hat.
refresh(p:Page) - Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.
deleteNews(ID:String) - Löscht die Neuigkeit mit der entsprechender ID.

1.7.2 NEUIGKEIT ERSTELLEN

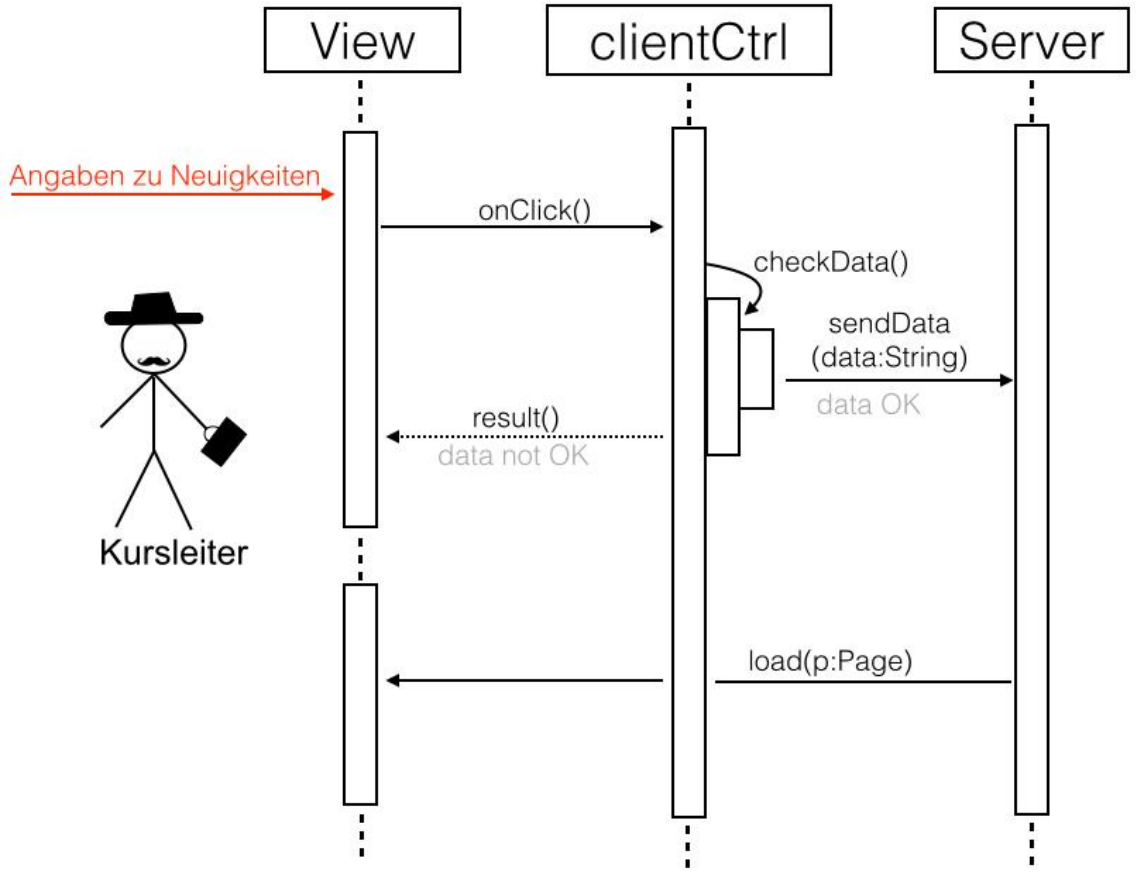

Abb. 16 Client Neuigkeit erstellen

Nachdem der Benutzer die Schaltfläche zum Erstellen einer neuen Neuigkeit betätigt hat, wird die Seite zur Eingabe dessen vom Server geladen. Danach kann der Sportleiter seine Nachricht für die anderen Benutzer eingeben. Wurde die Schaltfläche zum Bestätigen der Neuigkeit betätigt, so wird vom „clientCtrl“ kurz überprüft, ob überhaupt ein Text eingegeben wurde. Ist dies der Fall, so wird die Nachricht an den Server weiter geschickt und ins System aufgenommen. Im Anschluss wird die Startseite mit allen Neuigkeiten, ebenfalls mit der gerade eben erstellten, angezeigt. Sollte kein Text eingegeben worden sein, so erhält der Sportleiter eine kurze Meldung, dass die Neuigkeit verworfen wird.

onClick() - Der „clientCtrl“ wird aufgerufen.

sendData(data:String) - Die eingegebenen Daten werden an den Server übertragen.

checkData() - Nachdem der „clientCtrl“ sich die Daten aus dem View geholt hat, wird mit dieser Methode kurz überprüft, ob überhaupt ein Text eingegeben wurde. Wenn nicht, wird das Netzwerk nicht unnötig belastet und die Neuigkeit sofort verworfen. Der Nutzer erhält dann eine kurze Meldung. Wurde Text eingegeben, wird dieser an den Server weitergeleitet.

load(p:Page) - Die neue Seite wird geladen.

result() - Die Seite wird entsprechend der Fehler aktualisiert.

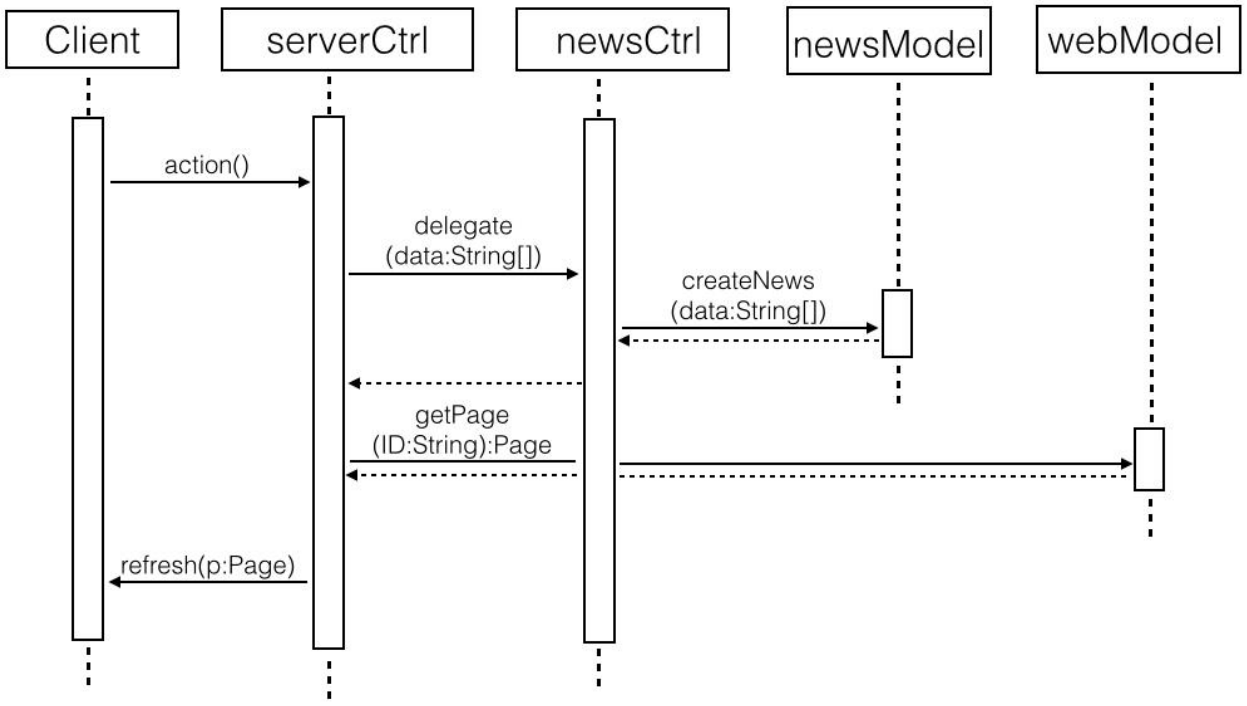

Abb. 17 Server Neuigkeit erstellen

Will der Kursleiter eine Neuigkeit erstellen, muss er erst alle Informationen eingeben. Nachdem die notwendigen Daten eingegeben wurden, erstellt der „newsCtrl“ die Neuigkeit („createNews(Data:String[])“). Danach wird dem Nutzer in einer neuen Website angezeigt, dass die Neuigkeit erfolgreich erstellt wurde (bezieht sich auf Spezifikation - Use-Case-Diagramm – Abb. 7).

action() - Übermittelt die Auswahl des Nutzers an den Server.

delegate(data:String[]) - Stößt den „subCtrl“ an und übergibt ihm alle notwendigen Daten als Argument.

createNews(data:String[]) – Erstellt eine neue Neuigkeit mit übergebenen Argumenten.

getPage(Id:String):Page - Liefert eine neue Webseite mit der entsprechenden ID zurück.

refresh(p:Page) – Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.

1.8 ALLGEMEINE FUNKTIONEN

1.8.1 BEWERTUNG DER KURSE

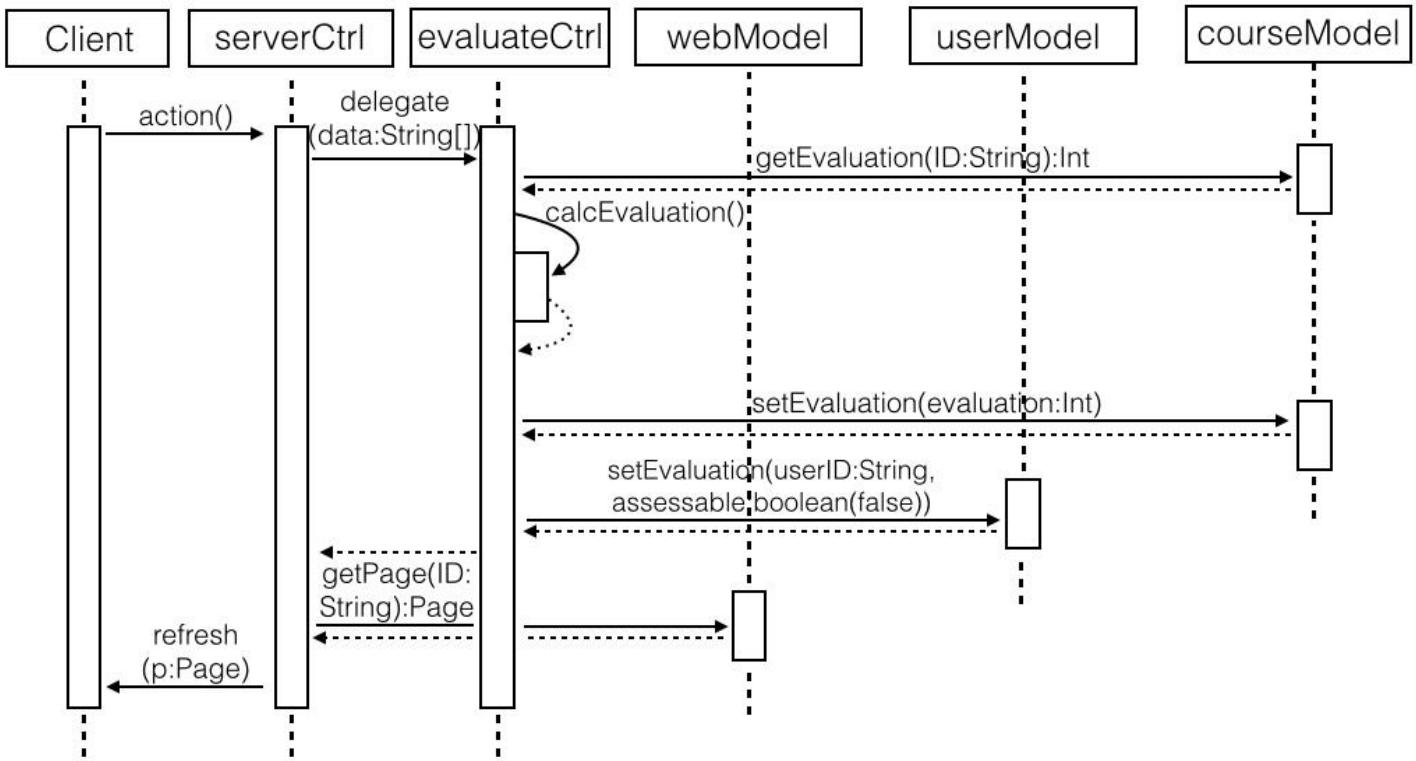

Abb. 18 Server Bewertung abgeben

Der Server-Controller empfängt die Daten vom Client und leitet diese nach einer Überarbeitung an den Subcontroller weiter. Dieser übernimmt das Verwerten der Daten. Nun wird mit Hilfe der KursID die Bewertung geladen. Diese und die eingegebene Bewertung werden mittels der Methode "calcEvaluation()" zusammengerechnet. Nach der Berechnung wird die neue Bewertung mit Hilfe der KursID wieder dem Kurs zugeordnet. Danach wird beim Nutzer unter Verwendung der Methode "setEvaluation(..)" im UserModel der Kurs als bewertet markiert. Danach wird eine neue Seite geladen.

action() - Liest die Daten aus dem Bewertungsformular aus und überträgt diese an den „serverCtrl“.
delegate(data:String[]) - Übermittelt die Daten vom „serverCtrl“ an den jeweiligen „subCtrl“.
getEvaluation(ID:String):Int - Holt die aktuelle Bewertung aus dem „courseModel“.
calcEvaluation() - Berechnet das Bewertungsergebnis aus dem alten Ergebnis im „courseModel“ und dem abgegebenen Ergebnis vom „Client“ und speichert dieses.
setEvaluation(evaluation:Int) - Schickt das neue Bewertungsergebnis zurück an das „courseModel“.
setEvaluation(userID:String, assessable:boolean(false)) - Dient dafür, dass der „Client“ keine weitere Bewertung zu diesem Kurs abgeben kann.
getPage(ID:String):Page - Liefert eine neue Webseite zurück.
refresh(p:Page) - Lädt die neue Seite, die der Nutzer im Weiteren zu sehen bekommt.

1.8.2 FUNKTIONSWEISE DER VORSCHLÄGE

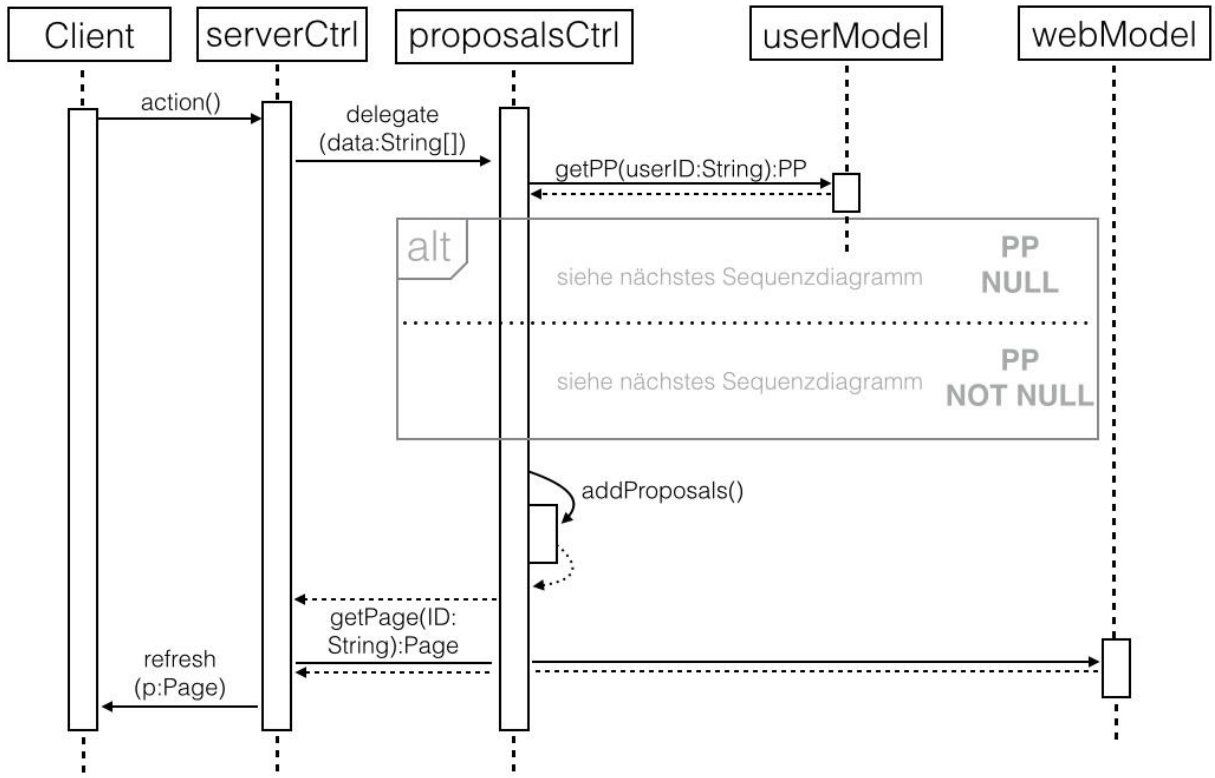

Abb. 19 Server Vorschläge einfügen in HTML-Seite

Sobald sich der Nutzer erfolgreich im OSM eingeloggt hat, wird ihm eine kleine Auswahl an Sportarten vorgeschlagen. Um diese Auswahl zu generieren, wird der „proposalsCtrl“ vom „serverCtrl“ durch die „delegate(data:String[])“ Methode aufgerufen. Das Leistungsprofil wird im Anschluss benötigt. Darum wird dieses über die „getPP(ID:Int):PP“-Methode aus dem „userModel“ aufgerufen, welches das Leistungsprofil des Nutzer mit der übergebenen ID zurückliefert. Ist das Leistungsprofil angekommen, so muss überprüft werden, ob dieses leer ist oder der Nutzer schon Kurse belegt hat. Diese Überprüfung findet in einer „If-Else-Anweisung“ statt, welche im nächsten Sequenzdiagramm ausführlich beschrieben wird. Durch die „addProposals“-Methode werden die generierten Vorschläge zur Seite hinzugefügt und über die „getPage(ID:String):Page“-Methode dem Nutzer kenntlich gemacht.

action() – Übermittelt die Auswahl des Nutzers an den Server.

delegate(data:String[]) - Übermittelt die Daten vom „serverCtrl“ an den jeweiligen „subCtrl“.

getPP(userID:String):PP - Liefert das Leistungsprofil des Nutzers mit der übergebenen ID aus dem userModel zurück.

addProposals() - Fügt die Vorschläge zur Seite hinzu.

getPage(ID:String):Page - Liefert eine neue Webseite zurück.

refresh(p:Page) - Lädt die neue Seite, die der Nutzer im weiteren zu sehen bekommt.

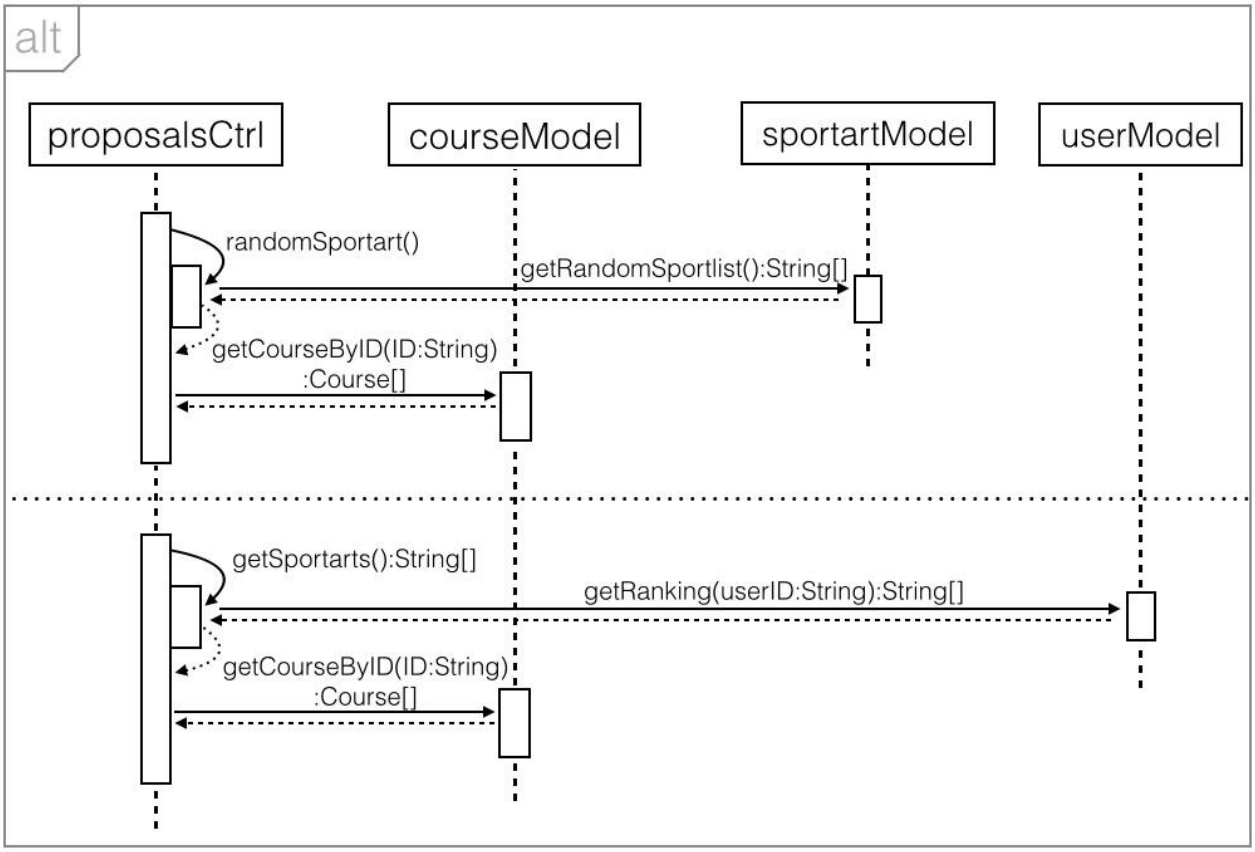

Abb. 20: Server Vorschläge If/else-Anweisung

Die „If-Else-Anweisung“ dient dazu, zwischen einem leeren und vollen Leistungsprofil zu unterscheiden. Ist dieses leer, so wird eine zufällig zusammengestellte Sportliste zurückgeliefert. Ist ein Leistungsprofil vorhanden, so werden die vorher generierten Vorschläge zurückgeliefert.

If:

In der If-Anweisung wird über die Methode „randomSportart()“ der Prozess angestoßen, in der im Weiteren über die Methode „getRandomSportlist():String[]“ eine zufällig erstellte Sportliste zurückgeliefert wird, die aus jeder Sportkategorie jeweils 2 Sportarten holt. Nach diesem Prozess werden die Kurse mit einer übergebenen ID über die „getCourseByID(ID:String):Course[]“-Methode aus dem „courseModel“ zurückgeliefert.

Else:

Ist ein Leistungsprofil vorhanden, so werden über die Methode „getSportarts():String[]“ alle Sportarten geholt, in der zunächst über die „getRanking(userID:String):String[]“-Methode die vorher generierten Vorschläge zurückgeliefert. Diese Sportarten werden mit den aktuell angebotenen Kursen verglichen, um den Vorschlägen die Kurse über die Methode „getCourseByID(ID:String):Course[]“ zuzuweisen.

randomSportart() - Stößt den Prozess an.
getRandomSportlist():String[] - Liefert eine zufällig erstellte Sportliste zurück.
getCourseByID(ID:String):Course[] - Liefert die Kurse mit der übergebenen ID zurück.
getSportarts():String[] - Liefert die Sportarten zurück.
getRanking(userID:String):String[] - Liefert das normale und defizit-Ranking zurück.
getCourseByID(ID:String):Course[] - Liefert die Kurse mit der übergebenen ID zurück.

1.8.3. VORSCHLÄGE GENERIEREN

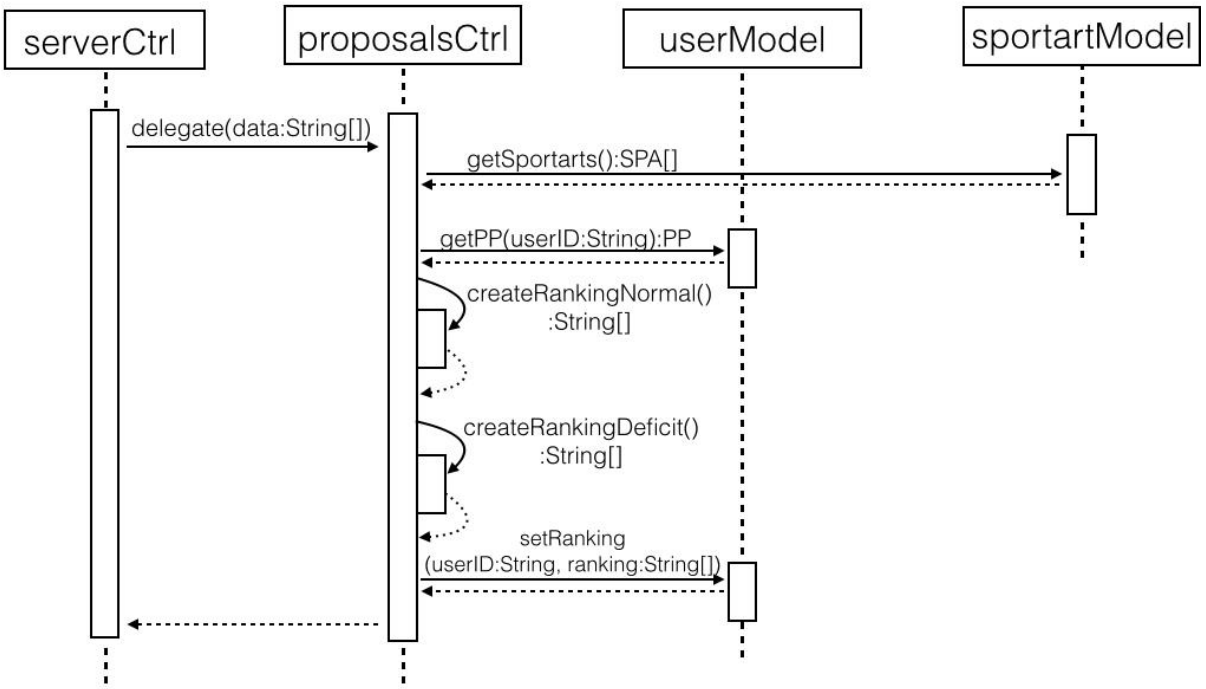

Abb. 21 Server Vorschläge generieren

Werden Sportvorschläge generiert, holt sich der subCtrl als erstes eine Liste aller Sportarten („getSportarts():SPA[]“) und das Leistungsprofil des Nutzers („getPP(ID:Int):PP“). Die Methoden „createRankingNormal():String[]“ und „createRankingDeficit():String[]“ generieren mit Hilfe der Sportarten und dem Leistungsprofil des Nutzers ein Ranking für die am besten passenden Sportarten bzw. für die Defizite. Danach werden die Rankings im „userModel“ gespeichert („setRankings(String[])“).

delegate(data:String[]): Stößt den subCtrl an und übergibt ihm alle notwendigen Daten als Argument.
getSportarts():SPA[] - Liefert die Liste mit allen Sportarten zurück.
getPP(userID:String):PP - Liefert das Leistungsprofil des Nutzers mit der übergebenen ID aus dem „userModel“ zurück:
createRankingNormal():String[] - Erstellt mit Hilfe des Leistungsprofils des Nutzers und den Leistungsprofilen der Kurse ein nutzerspezifisches Ranking für die am besten passenden Sportarten.

createRankingDeficit():String[] - Erstellt mit Hilfe des Leistungsprofils des Nutzers und den Leistungsprofilen von den Kursen ein nutzerspezifisches Ranking für die Sportarten, die die Defizite des Nutzers ausgleichen.

setRanking(userID:String, ranking:String[]) - Speichert die Rankings im „userModel“ des Nutzers.

1.8.4 LEISTUNGSPROFIL AKTUALISIEREN

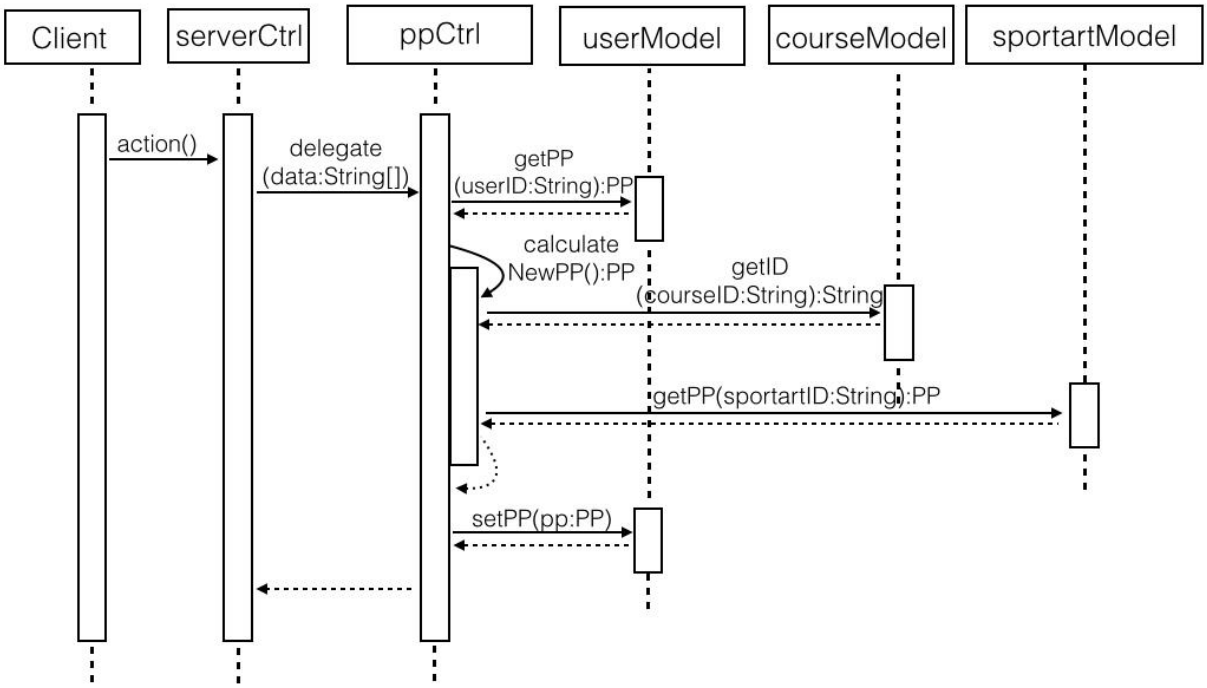

Abb. 22 Server Leistungsprofil aktualisieren

Wenn das Leistungsprofil eines Nutzers aktualisiert wird, holt sich der „ppCtrl“ als erstes das aktuelle Leistungsprofil des Nutzers aus dem „userModel“ über die Methode „getPP(userId:String):PP“. Danach wird die Methode „calculateNewPP():PP“ aufgerufen, welche die Werte des neuen Leistungsprofils berechnet. Um die Werte berechnen zu können, holt sich der subCtrl mit der Methode „getID(courseID:String):String“ die ID des belegten Kurses aus dem „courseModel“. Mit dieser ID kann sich der subCtrl aus dem „sportartModel“ das Leistungsprofil des belegten Kurses holen („getPP(sportartID:String):PP“). Danach wird das Leistungsprofil des Nutzers mit dem gerade geholten Leistungsprofil verrechnet, und das neu errechnete Leistungsprofil wird über die Methode „setPP(pp:PP)“ im userModel gespeichert.

action() - Übermittelt die Auswahl des Nutzers an den Server.

delegate(data:String[]): Stößt den subCtrl an und übergibt ihm alle notwendigen Daten als Argument.

getPP(userID:String):PP - Liefert das Leistungsprofil des Nutzers mit der übergebenen ID aus dem „userModel“ zurück.

calculateNewPP():PP - Berechnet ein neues Leistungsprofil, welches aus den alten Werten aus dem „userModel“ und den neuen Werten berechnet wird.

getID(courseID:String):String - Liefert die ID des vom Nutzer belegten Kurses zurück.

getPP(sportartID:String):PP - Liefert das Leistungsprofil des vom Nutzer belegten Kurses mithilfe der zuvor geholten ID zurück.

setPP(pp:PP) - Speichert das neu errechnete Leistungsprofil vom Nutzer im „userModel“.

2 KLASSENDIAGRAMM

2.1 ÜBERSICHT KLASSENDIAGRAMM

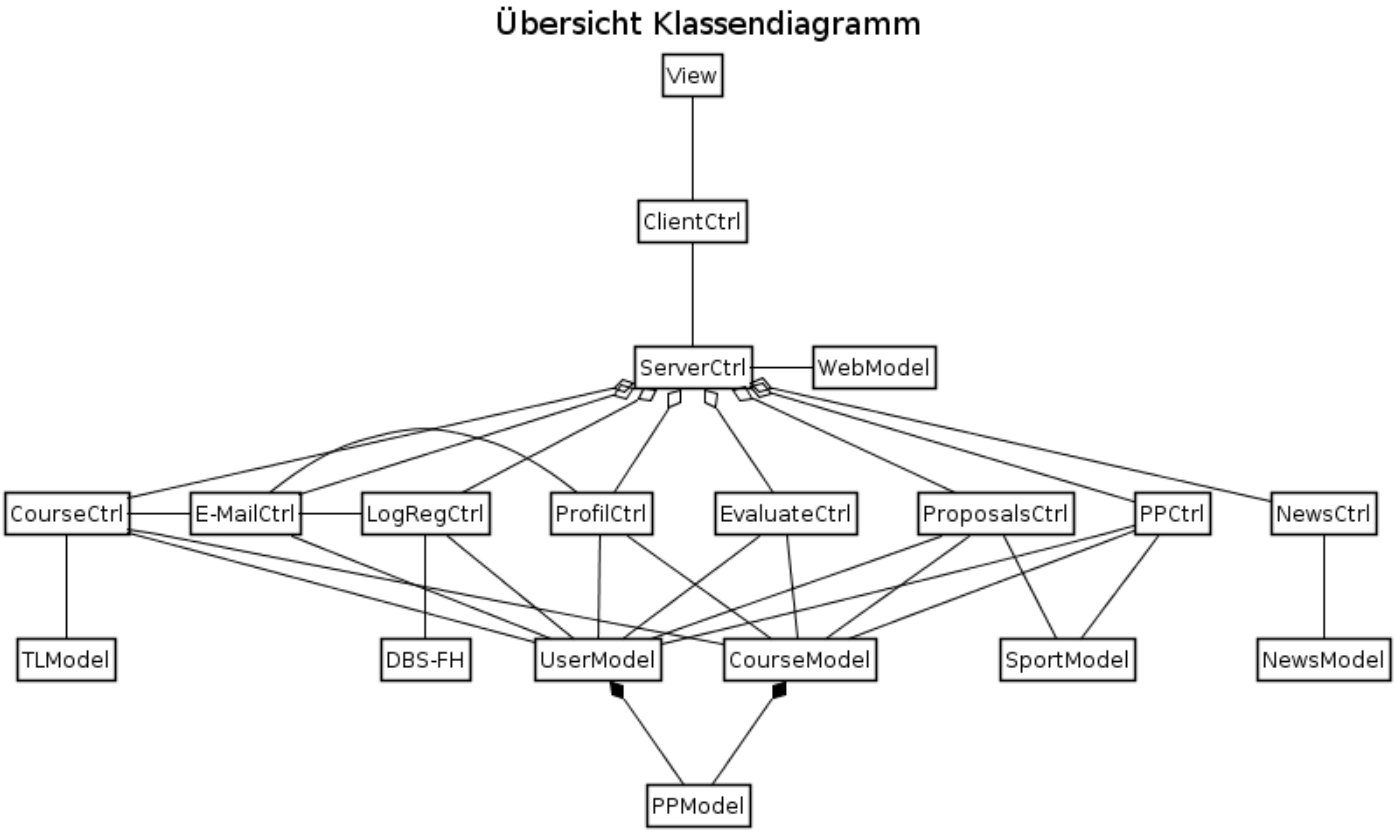

Abb. 23 Klassendiagramm Übersicht

2.2 ÜBERSICHT METHODEN DER KLASSEN

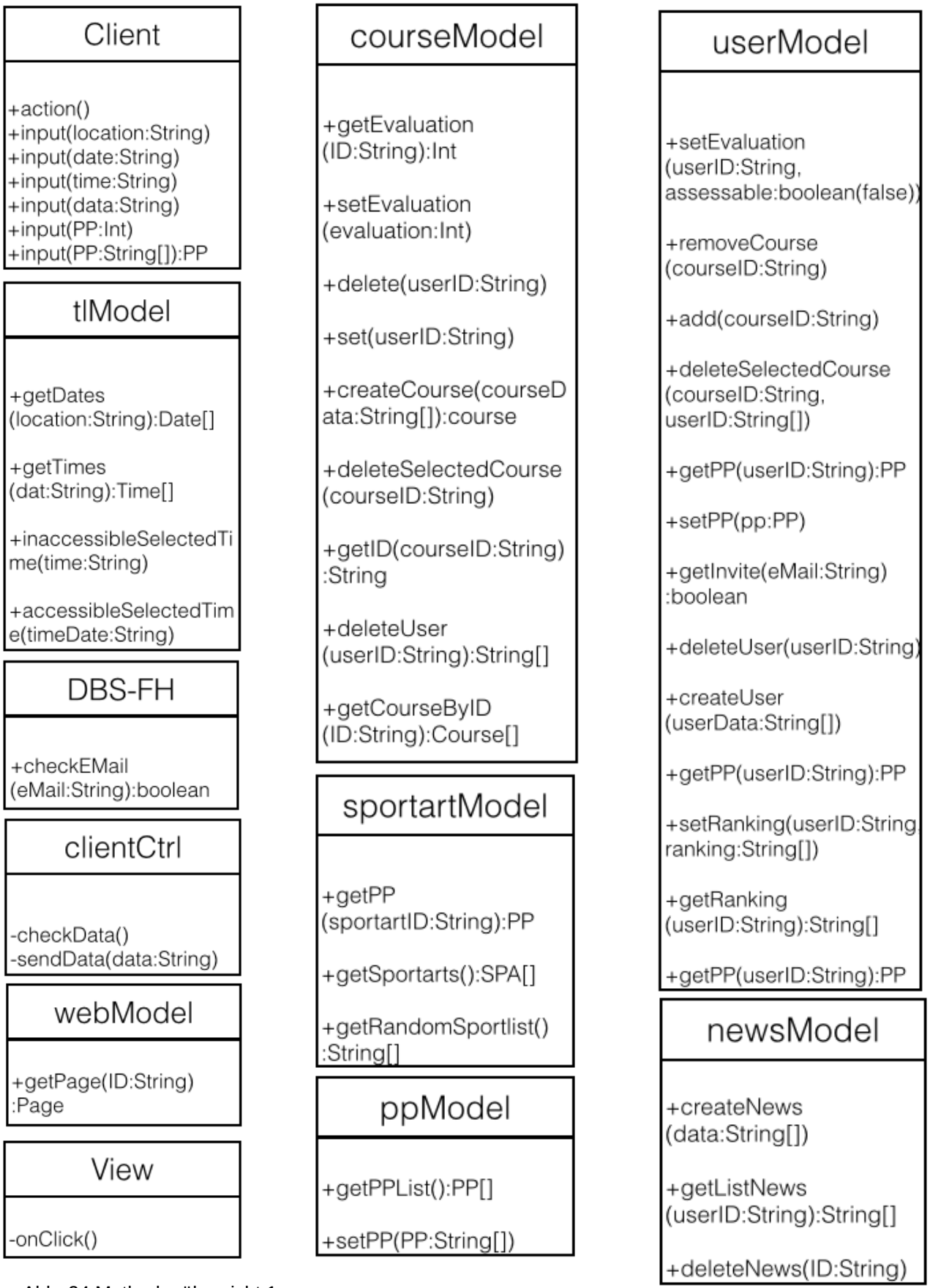

Abb. 24 Methodenübersicht 1

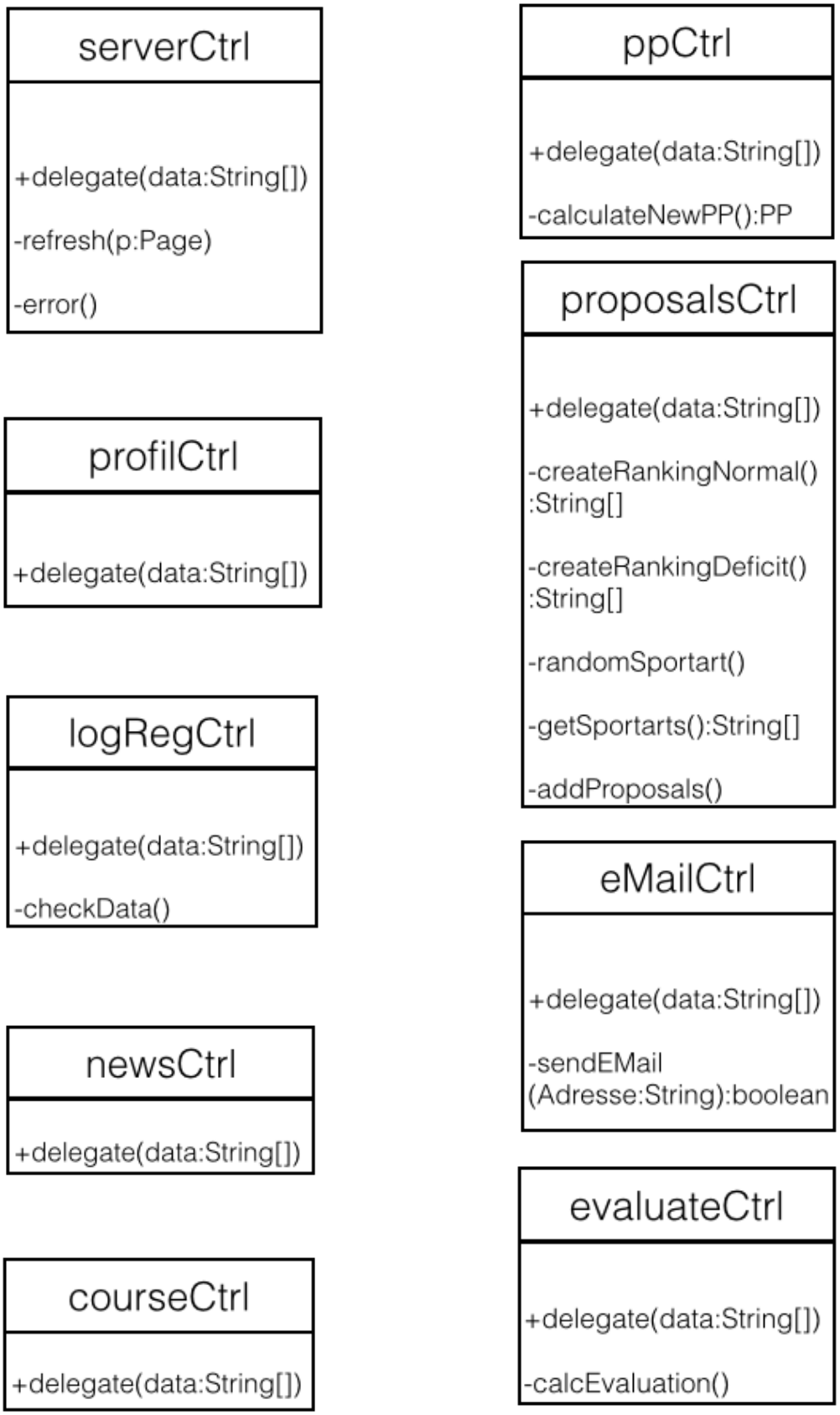

Abb. 25 Methodenübersicht 2