UE Software Engineering 050052 – Gruppe 10
WS 2015/16

LV-Leiter: Hans Moritsch

Designmodell II

Projektname: Blue Couch - Das soziale Netzwerk

Projektteam:

Nachname	Vorname	Matrikelnummer	E-Mail-Adresse
Gazar	Mohamed	a0928951	a0928951@unet.univie.ac.at
Kolhaupt	Raphael	a1407523	a1407523@unet.univie.ac.at
Misurec	Patrik	a1325267	a1325267@unet.univie.ac.at
Pfneisl	Christian	a9525708	a9525708@unet.univie.ac.at

CEWebS-Teamseite: Team 6: SN2 - Blue Couch

Datum: 01.Jan.16

Inhalt:

1 Klassendesign

Beginnend folgt eine Beschreibung der wichtigsten Klassen sowie ihre Methoden und Attribute.

1.1. Klasse Anwender (abstract)

Basisklasse für die unterschiedlichen Anwendertypen. Subklassen sind User, Forscher, und Admin. Stellt Funktionen zur Verfügung, die jede dieser Subklassen benötigt.

Attribute

- eMailID : String
    - eine eindeutige ID jedes Anwenders (bestehend aus seiner, beim LogIn angegebenen eMail. Hiermit wird der Anwender eindeutig Identifiziert, um unter anderem zu entscheiden in welche Rolle (Admin, Forscher, User) der Anwender fällt (stellt ihm die jeweiligen Methoden zur Verfügung)
- Password : String
    - Das Passwort des Anwenders, um auf Zugriff auf seine Daten vor anderen zu schützen. Dient neben der eMailID als zusätzliche Authentifikationsmaßnahme
- Erstellt : String
    - Das Registrierungsdatum des Anwenders

1.2. Klasse Admin (extends Anwender)

Klasse die der verwaltung anderer Nutzer und deren Beiträge dient. Erweitert Basisklasse Anwender um „Admin“ Funktionen und Attribute. Erhält außerdem Benachrichtungen bezüglich von Usern gemeldeten Beiträgen mittels eigener Klassenvariable.

Attribute

- gemeldeteBeitraege : ArrayList<int>
    - Liste aller gemeldeten Beiträge – für jeden Admin gleich. Wenn ein Admin einen Beitrag loescht / freigibt → wird auch aus Liste gemeldeter Beitraege geloescht

Methoden

- meldungAufheben(beitrag : Beitrag) : void
    - Dient dem Aufheben einer Meldung eines gemeldeten Beitrages und führt schlussendlich zum Löschen eben jenes aus der Instanzvariable gemeldeteBeitraege
- loescheBeitrag(id : int, user : User) : void
    - Dient dem Löschen eines zuvor gemeldeten Beitrages und führt schlussendlich zum Löschen eben jenes aus der Instanzvariable sowie aus der Instanzvariable beitraege des jeweiligen Benutzers vom Typ „User“.
- userSperren(user : User, endDatum : Calendar) : void
    - Dient dem sperren eines Users, dessen Beitrag zuvor von einem anderen User gemeldet wurde. Hierbei gilt: endDatum == null führt zu einer dauerhaften Sperre, andernfalls lediglich zu einer Sperre bis zum Wert der Variable endDatum.

1.3. Klasse Forscher (extends Anwender)

Klasse Forscher, Erweiterung der Basisklasse Anwender, beinhaltet Methoden zur statistischen Auswertung des aktuellen Zustands.

Methoden

- anzahl_beitraege_durchschnitt(userlist : ArrayList<Anwender>) : double
    - Berechnet die durchschnittliche Anzahl an Beiträgen pro User. Es werden alle Anwender ausschließlich User aus der übergebenen Liste herausgefiltert, und anschließend die Anzahl der Beiträge der restlichen Anwender aufsummiert sowie durch die Anzahl der restlichen Anwender dividiert.
- anzahl_beitraege_monat(userlist : ArrayList<Anwender>) : double
    - Berechnet die Anzahl an neu verfassten Beiträge pro User innerhalb des letzten Monats. Filtert aus der übergebenen Liste sämtliche Objekte die nicht vom Typ „User“ sind heraus, und summiert bei den übrigen die Anzahl der Beiträge welche innerhalb des letzten Monats verfasst wurden auf und dividiert sie anschließend durch die Anzahl der übrig gebliebenen Anwender.
- anzahl_freundschaften_durchschnitt(userlist : ArrayList<Anwender>) : double
    - Berechnet die durchschnittliche Anzahl an Freundschaften pro User. Es werden alle Anwender ausschließlich User aus der übergebenen Liste herausgefiltert, und anschließend die Anzahl der Freundschaften der restlichen Anwender aufsummiert sowie durch die Anzahl der restlichen Anwender dividiert.
- anzahl_freundschaften_monat(userlist : ArrayList<Anwender>) : double
    - Berechnet die Anzahl an neu geschlossenen Freundschaften pro User innerhalb des letzten Monats. Filtert aus der übergebenen Liste sämtliche Objekte die nicht vom Typ „User“ sind heraus, und summiert bei den übrigen die Anzahl der Freundschaften welche innerhalb des letzten Monats geschlossen wurden auf und dividiert sie anschließend durch die Anzahl der übrig gebliebenen Anwender.
- anzahl_freundschaften(userlist : ArrayList<Anwender>) : double
    - Berechnet die gesamte Anzahl an Freundschafte. Es werden alle Anwender ausschließlich User aus der übergebenen Liste herausgefiltert, und anschließend die Anzahl der Freundschaften der restlichen Anwender aufsummiert und schlussendlich durch 2 geteilt (da ja jede Freundschaftsbeziehung 2x vorkommt).
- anzahl_sperren(userlist : ArrayList<Anwender>) : double
    - Berechnet die gesamte Anzahl an aktuellen Sperren. Es werden alle Anwender ausschließlich User aus der übergebenen Liste herausgefiltert, und anschließend die Anzahl der Sperren der restlichen Anwender aufsummiert.

1.4. Klasse User (extends Anwender)

„Herzstück“ des Projektes – Erweiterung der Basisklasse Anwender – stellt eigentliche Möglichkeit zur Verfügung User, samt aller Attribute anzulegen und zu verwalten. User legt seinerseits eigene Klassen zum Ausbau an.

Attribute

- SichtbarkeitFreundesListe : ArrayList<Gruppe>
    - Eine Liste der Gruppen dessen Mitglieder die Berechtigung besitzen die entsprechende FreundesListe des Users zu sehen.
- SichtbarkeitPinnwand : ArrayList<Gruppe>
    - Eine Liste der Gruppen dessen Mitglieder die Berechtigung besitzen die entsprechende Pinnwand des Users zu sehen.
- SichtbarkeitKontaktdaten : ArrayList<Gruppe>
    - Eine Liste der Gruppen dessen Mitglieder die Berechtigung besitzen die entsprechende Kontaktdaten des Users zu sehen.
- SichtbarkeitProfilInfo : ArrayList<Gruppe>
    - Eine Liste der Gruppen dessen Mitglieder die Berechtigung besitzen die entsprechende Profilinfo des Users zu sehen.
- gesperrt : boolean = false
    - Wert der anzeigt, ob ein User aktuell eine Sperre besitzt
- gesperrtBis : Calendar
    - Wert der anzeigt, wie lange ein User gesperrt ist.

[Anmerkung: 3 Kombinationen; gesperrt false, gesperrt bis Null → User nicht gesperrt, gesperrt true, gesperrt bis Calendar → User bis Datum gesperrt, gesperrt true, gesperrt bis Null → User dauerhaft gesperrt]

- SperreNR : int = 0
    - Gibt an wie oft ein User bereits gesperrt wurde
- kontaktdaten : Kontaktdaten
    - Beinhaltet das Objekt kontaktdaten vom typ Kontaktdaten, welche eindeutig dem jeweiligen User zugeordnet ist (beidseitige Absicherung)
- profilinfo : Profilinfo
    - Beinhaltet das Objekt profilinfo vom typ Profilinfo, welche eindeutig dem jeweiligen User zugeordnet ist (beidseitige Absicherung)

- beitraege : ArrayList<Beitrag>
    - Beinhaltet die Objekte beitraege vom typ Beitrag, welche eindeutig dem jeweiligen User zugeordnet ist (beidseitige Absicherung)
- freunde : ArrayList<Freund>
    - Beinhaltet die Objekte freunde vom typ Freund, welche eindeutig dem jeweiligen User zugeordnet ist (beidseitige Absicherung)
- nutzerbild : String
    - Beinhaltet den Namen des Nutzerbilds des Nutzers
- gruppen : ArrayList<Gruppe>
    - Beinhaltet die Objekte gruppen vom typ Gruppe, welche eindeutig dem jeweiligen User zugeordnet ist (beidseitige Absicherung)
- freundschaftsanfragen_ausgehend : ArrayList<String>
    - Beinhaltet alle getaetigten ausgehenden Freundschaftsanfragen des Users
- freundschaftsanfragen_eingehend : ArrayList<String>
    - Beinhaltet alle eingehenden Freundschaftsanfragen des Users die dieser beantworten muss

Methoden

- anzahlFreunde() : int
    - Methode um die Anzahl der gefundenen Freunde eines Users zu bekommen. Nimmt Instanzvariable freunde zur Hilfe.
- anzahlFreundeZeit(zeit : Calendar) : int
    - Methode um die Anzahl der geschlossenen Freundschaften eines Users innerhalb einer Zeitspanne zu bekommen. Nimmt Instanzvariable freunde zur Hilfe.
- freundLoeschen(name : String) : void
    - Methode um eine Freundschafts bezüglich eines anderen Nutzers zu entfernen. Dieser Nutzer wird aus der Liste freunde entfernt. Erfolgt auf beidseitigem Niveau → Auch Instanzvariable des anderen Nutzers wird entfernt
- freundschaftsanfrageVersenden(name : String) : void
    - Methode um eine Freundschaftsanfrage an einen anderen Nutzer zu versenden. Dieser erhält eine Freundschaftseinladung (Speicherung der Einladung in der Instanzvariable freundschaftsanfragen_ausgehend). Dieser Nutzer wird der Instanzvariable freundschaftsanfragen_ausgehend des aktuellen „einladenden“ Nutzers hinzugefügt.

- freundschaftsanfrageBeantworten(name : String) : void
    - Methode um eine eingegangene Freundschaftsanfrage zu beantworten. Hierbei besteht entweder die Möglichkeit diese abzulehnen aka zu verwerfen (i.e. sie aus der Instanzvariable freundschaftsanfragen_eingehend zu löschen) oder sie anzunehmen, woraufhin sie zusätzlich zum löschen innerhalb der Instanzvariable freundschaftsanfragen_eingehend, auch mittels Erzeugung eines neuen Objektes vom Typ Freund und dem hinzufügen zur Instanzvariable freunde gespeichert wird.
- beitragLoeschen(id : int) : void
    - Methode um einen verfassten Beitrag wieder zu loeschen. Hierbei wird die Verbindung zwischen dem jeweiligen Beitragsobjekts und dem User entfernt, um dieses zu löschen.
- anzahlBeitrage() : int
    - liefert die aktuelle Anzahl der (nicht geloeschten) Beiträge zurück.
- anzahlBeitraegeZeit(help : Calendar) : int
    - liefert die Anzahl der (nicht geloeschten) Beiträge zurück, welche innerhalb einer bestimmten Zeitspanne verfasst worden sind.
- beitragHinzufuegen(titel : String, inhalt : String) : void
    - Erstellt ein neues Objekt vom Typ Beitrag und fuegt es der Instanzvariable beitraege hinzu. Dieses neue Objekt wird mit entsprechendem Erstelldatum, übergebenem Titel und übergebenen (Inhalts-)Text versehen.
- gruppeErstellen(name : String) : void
    - Der User erhält die Möglichkeit ein neues Objekt vom Typ Gruppe zu erstellen, welches Standartmäßig einen Namen erhält. Weiters wird dieses erzeugte Objekt der Instanzvariable gruppen des Users angefügt.
- gruppeLoeschen(name : String) : void
    - Der User erhält die Möglichkeit ein Objekt vom Typ Gruppe zu loeschen. Kaskadierend wird die Instanzvariable der in der jeweiligen Gruppe angefügten Freunde in_gruppe auf null gesetzt. Das zu loeschende Objekt wird anschließend aus der Instanzvariable gruppen des Users entfernt.
- switch gesperrt() : void
    - Wenn User gesperrt ist -> Entsperrt den User. Invertiert andernfalls.
- addSichtbarkeitFreundesliste(name : String) : void
    - Fügt der Instanzvariable sichtbarkeitFreundesliste(ArrayList<Gruppe>) eine neue Gruppe hinzu (Welche den Namen „name“ trägt). Sollte diese bereits vorhanden sein wird hingegen nichts verändert.

- addSichtbarkeitPinnwand(name : String) : void
    - Fügt der Instanzvariable sichtbarkeitPinnwand(ArrayList<Gruppe>) eine neue Gruppe hinzu (Welche den Namen „name“ trägt). Sollte diese bereits vorhanden sein wird hingegen nichts verändert.
- addSichtbarkeitKontaktdaten(name : String) : void
    - Fügt der Instanzvariable sichtbarkeitKontaktdaten(ArrayList<Gruppe>) eine neue Gruppe hinzu (Welche den Namen „name“ trägt). Sollte diese bereits vorhanden sein wird hingegen nichts verändert.
- addSichtbarkeitProfilinfo(name : String) : void
    - Fügt der Instanzvariable sichtbarkeitProfilinfo(ArrayList<Gruppe>) eine neue Gruppe hinzu (Welche den Namen „name“ trägt). Sollte diese bereits vorhanden sein wird hingegen nichts verändert.

1.5. Klasse Profilinfo

Die Klasse Profilinfo beinhaltet alle wesentlichen Attribute des Users bezüglich seines Profils. Sie ist einem User eindeutig anhand dessen emailID zugewiesen, wobei jeder User genau ein Objekt der Klasse Profilinfo „besitzt“.

Attribute

- vorname : String
    - Der Vorname des Users
- nachname : String
    - Der Nachname des Users
- geburtsdatum : Calendar
    - Das Geburtsdatum des Users
- hobbys : String
    - Die Hobbies des Users
- geschlecht : String
    - Das Geschlecht des Users
- interessen : String
    - Die Interessen des Users
- religion : String
    - Die Religion des Users
- familienstatus : String
    - Der Familienstatus des Users
- sexOrientierung: String
    - Die sexuelle Orientierung des Users
- polEinstellung: String
    - Die politische Einstellung des Users
- zusatzangaben : String
    - Zusatzangaben, welche der User tätigen will
- user : String
    - Der User zugehörig zum jeweiligen Objekt vom Typ Profilinfo ist hier gespeichert.

Methoden

- setAll(vorname : String, nachname : String, geburtsdatum : Calendar, hobbies : String, geschlecht : String, religion : String, interessen : String, sexOrientierung : String, polEinstellung : String, zusatzangaben : String) : void

    - Dient der gemeinsamen Änderung der gesamten Profilinfo (ohne separate alle getX Anfragen versenden zu müssen)

1.6. Klasse Kontaktdaten

Die Klasse Kontaktdaten beinhaltet alle wesentlichen Attribute des Users bezüglich seiner Kontaktdaten. Sie ist einem User eindeutig anhand dessen emailID zugewiesen, wobei jeder User genau ein Objekt der Klasse Kontaktdaten „besitzt“.

Attribute

- adresse : String
    - Die Adresse des Users
- wohnort : String
    - Der Wohnort des Users
- telefonnr : String
    - Die Telefonnummer des Users
- email : String
    - Die eMail des Users
- user : String
    - Der User zugehörig zum jeweiligen Objekt vom Typ Kontaktdaten

Methoden

- setAll(adresse : String, wohnort : String, telefonnr : String, eMail : String) : void
    - Dient der gemeinsamen Änderung der gesamten Kontaktdaten (ohne separate alle getX Anfragen versenden zu müssen)

1.7. Klasse Gruppe

Die Klasse Gruppe beinhaltet alle wesentlichen Attribute des Users bezüglich seiner Gruppen. Sie ist einem User eindeutig anhand dessen emailID zugewiesen, wobei jeder User mehere Objekte der Klasse Gruppe besitzen kann.

Attribute

- name : String
    - Der Name der Gruppe
- mitglieder : ArrayList<Freund>
    - Die Mitglieder der Gruppe (Referenz auf jeweiliges Gruppenobjekt)
- user : String
    - Der User zugehörig zu jeweiligen Objekten vom Typ Gruppe

Methoden

- freund_zu_gruppe(freund : Freund) : void
    - Eine Referenz vom Typ Freund zur Instanzvariable mitglieder der Gruppe hinzufügen. Das Objekt vom Typ Freund bekommt weiterhin einen Vermerk innerhalb seiner Instanzvariablen, zu welcher Gruppe es zugeordnet wurde
- freund_aus_gruppe(freund : Freund) : void
    - Eine Referenz aus der Instanzvariable mitglieder der Gruppe entfernen. Das Objekt vom Typ Freund bekommt weiterhin einen Vermerk innerhalb seiner Instanzvariablen, zu welcher Gruppe es zugeordnet wurde

1.8. Klasse Freund

Die Klasse Freund beinhaltet alle wesentlichen Attribute des Users bezüglich seiner Freunde. Sie ist einem User eindeutig anhand dessen emailID zugewiesen, wobei jeder User mehere Objekte der Klasse Freund besitzen kann.

Attribute

- beginn : Calender
    - Erstellungsdatum des Objekts vom Typ Freund
- in_gruppe : Gruppe
    - Vermerk, zu welcher Gruppe der jeweilige Freund zugeordnet wurde
- user : String
    - Der User zugehörig zu jeweiligen Objekten vom Typ Gruppe
- freunduser : String
    - Der User mit dem die Freundschaft eingegangen wurde

1.9. Klasse Beitrag

Die Klasse Beitrag beinhaltet alle wesentlichen Attribute des Users bezüglich seiner Beiträge. Sie ist einem User eindeutig anhand dessen emailID zugewiesen, wobei jeder User mehere Objekte der Klasse Beitrag besitzen kann.

Attribute

- user : String
    - Der User zugehörig zu jeweiligen Objekten vom Typ Beitrag
- titel : String
    - Titel des Beitrages
- erstellt : Calender> = today
    - Datum an dem der Beitrag erstellt wurde
- inhalt : String
    - Textueller Inhalt des jeweiligen Beitrages
- gemeldet : boolean
    - Wert der angibt ob jeweiliger Beitrag gemeldet wurde
- id : int
    - Die eindeutige BeitragsID des jeweiligen Beitrags. Fortlaufend inkrementierende Nummer, welche bei Erstellung eines Beitrag hochgezählt wird.

1.10. Klasse AnwenderManagement

Die Klasse AnwenderManagement regelt die interne Logik, sowie die weiterleitung der Befehle die die Servlets benötigen. Es überprüft ob Aktionen erlaubt sind, oder nicht, aka führt diese aus oder verweigert sie – dient also quasi gleichzeitig als Sicherheitssystem (zumindest von der Logik-Seite aus)

Attribute

- anwenderDAO : AnwenderDAO
    - AnwenderManagement benötigt Hilfe von AnwenderDAO, da diese für das persistente Verwalten der Daten verantwortlich ist, weshalb AnwenderManagement eine Referenz auf ein Objekt des Typs anwenderDAO erhaelt (bzw. genauer gesagt, ein Objekt vom Typ anwenderDAO erzeugt)

Methoden

- getAnwenderbyID(email : String) : Anwender
    - Methode getAnwenderbyID retourniert unter Zuhilfenahme der Klasse AnwenderDAO den gefundenen Anwender. Sollte kein Anwender gefunden worden sein, erfolgt eine Rückgabe von null.
- userSperrenAdmin(emailUser : String, emailAdmin : String, cal : Calendar) : void
    - Methode userSperrenAdmin sperrt einen User unter Zuhilfename der Klasse Admin.java.
- istGesperrt(emailUser : String) : Boolean
    - Methode istGesperrt überprüft ob der User gesperrt ist.
- getGemeldeteBeitraege(emailAdmin : String) : ArrayList<Beitrag>
    - Methode getGemeldeteBeitraege gibt alle gemeldeten Beiträge unter Zuhilfenahme der Klasse Admin zurück.
- beitragFreigebenAdmin(emailUser : String, id : int) : void
    - Methode beitragFreigebenAdmin gibt einen gemeldeten Beitrag unter Zuhilfenahme der Klasse Admin.java frei.
- beitragLeoschenAdmin(emailUser : String, id : int) : void
    - Methode beitragLoeschenAdmin löscht einen gemeldeten Beitrag unter Zuhilfenahme der Klasse Admin.java sowie der Klasse AnwenderManagement.java.
- login(email : String, password : String) : String
    - Der Anwender muss sich vor dem ausführen zuerst mittels Methode LogIn identifizieren. Diese überprüft, unter Zuhilfename der Methoden des AnwenderDAO's ob der Anwender registriert ist, und um welchen Typ Anwender (g.e. User, Admin, Forscher) es sich handelt, und gibt Rückmeldung über ein erfolgreiches oder negatives Anmelden.

- freundStorno(email : String, freundmail : String) : void
    - freundStorno storniert eine Freundschaftsanfrage des Users. Löscht diese Anfrage sowohl aus den eingehenden Freundschaftsanfragen des jeweils noch beteiligten Users als auch aus den ausgehenden Freundschaftsanfragen des versendenden Users.
- freundAdd(email : String), freundmail : String) : void
    - freundAdd versendet eine Freundschaftsanfrage an den zu addenden User. Diese Anfrage wird sowohl in den eingehenden Freundschaftsanfragen des jeweils noch beteiligten Users, als auch in den ausgehenden Freundschaftsanfragen des versendenden Users hinzugefügt.
- freundDel(email : String, freundmail : String) : void
    - freundDel loescht einen Freund aus den bestehenden Freunden des Benutzers. Diese Freundschaft wird bei beiden Usern entfernt!
- freundschaftNein(email : String, freundmail : String) : void
    - freundschaftNein loescht eine eingehende Freundschaftsanfrage. Löscht diese Anfrage sowohl aus den eingehenden Freundschaftsanfragen des Users als auch aus den ausgehenden Freundschaftsanfragen des jeweils anderen Users.
- freundschaftJa(email : String, freundmail : String) : void
    - freundschaftJa akzeptiert eine eingehende Freundschaftsanfrage positiv. Löscht diese Anfrage sowohl aus den eingehenden Freundschaftsanfragen des Users als auch aus den ausgehenden Freundschaftsanfragen des jeweils anderen Users und fügt stattdessen bei beiden Usern die jeweilige Freundschafts hinzu.
- registrieren(email : String, password : String) : String
    - Der Benutzer muss sich bei seinem ersten Besuch registrieren, wobei ein neues Objekt vom Typ User/Anwender/Forscher angelegt wird. Hierbei muss eine eindeutige eMailID (welche noch nicht durch einen anderen User belegt sein darf) sowie ein Passwort benötigt. Anschließend wird der neu angelegte Benutzer mittels Methoden des AnwenderDAO's persistent gespeichert.
- gruppeSichtProfilinfo(email : String, gruppennamen : String) : boolean
    - gruppeSichtProfilinfo überprueft ob die Gruppe mit dem Titel gruppennamen die Berechtigung die Profilinfo zu sehen.
- gruppeSichtKontaktdaten(email : String, gruppennamen : String) : boolean
    - gruppeSichtKontaktdaten überprueft ob die Gruppe mit dem Titel gruppennamen die Berechtigung hat die Kontaktdaten zu sehen.
- gruppeSichtPinnwand(email : String, gruppennamen : String) : boolean
    - gruppeSichtPinnwand überprueft ob die Gruppe mit dem Titel gruppennamen die Berechtigung hat die Pinnwand zu sehen.

- gruppeSichtFreundesliste(email : String, gruppennamen : String) : boolean
    - gruppeSichtFreundesliste überprueft ob die Gruppe mit dem Titel gruppennamen die Berechtigung hat die Freundesliste zu sehen.
- getGruppen(email : String) : ArrayList<String>
    - getGruppen gibt eine ArrayList der Gruppen des Users zurueck.
- getFreundeID(email : String) : ArrayList<String>
    - getFreundeID gibt eine ArrayList der Freunde des Users zurueck.
- getFAnfragenEingehend(email : String) : ArrayList<String>
    - getFAnfragenEingehend gibt die eingehenden Freundschaftsanfragen zurueck.
- getFAnfragenAusgehend(email : String) : ArrayList<String>
    - getFAnfragenAusgehend gibt die ausgehenden Freundschaftsanfragen zurueck.
- getBeitragsMeldung(email : String) : ArrayList<Boolean>
    - getBeitragMeldungen gibt eine ArrayList mit den jeweiligen MeldeStati der Beitraege des angegebenen Users zurueck.
- getBeitraegsID(email : String) : ArrayList<Integer>
    - getBeitraegsID gibt alle Beitrags IDs des Users als ArrayList zurueck.
- getBeitraegsInhalt(email : String) : ArrayList<String>
    - getBeitraegsInhalt gibt den gesamten Inhalt aller Beitraege eines User als ArrayList zurueck.
- getBeitraegsTitel(email : String) : ArrayList<String>
    - getBeitraegsTitel gibt die Titel aller Beitraege eines Users als ArrayList zurueck.
- getBenutzerbildFromUser(email : String) : String
    - getBenutzerbildFromUser gibt das File des Benutzerbildes zurueck.
- getBenutzerbildFuerUser(email : String, filename : String) : void
    - setBenutzerbildFuerUser fuegt den Pfad des Benutzerbildes in der Instanzvariable des Users hinzu.
- getBday(email : String) : String
    - getBday gibt das Geburtsdatum eines Users zurueck im Datumsformat yyyy/MM/dd.

- getNachname(email : String) : String
    - getNachname gibt den Nachnamen eines Users zurueck.
- anzahlBeitraege(email : String) : int
    - anzahlBeitraege gibt die Anzahl der erstellten Beitraege eines Users zurueck.
- getAccoutnEmail(email : String) : String
    - getAccountEmail gibt die emailID eines Users zurueck.
- getEmail(email : String) : String
    - getEmail gibt die Email der Kontaktdaten eines Users zurueck.
- getTelNr(email : String) : String
    - getTelNr gibt die Telefonnummer eines Users zurueck.
- getWohnort(email : String) : String
    - getWohnort gibt den Wohnort eines Users zurueck.
- getAdresse(email : String) : String
    - getAdresse gibt die Adresse eines Users zurueck.
- getZusatzangaben(email : String) : String
    - getZusatzangaben gibt die Zusatzangaben eines Users zurueck.
- getPolEinstellung(email : String) : String
    - getPolEinstellung gibt die politische Einstellung eines Users zurueck.
- getSexOrientierung(email : String) : String
    - getSexOrientierung gibt die sexuelle Orientierung eines Users zurueck.
- getFamilienstatus(email : String) : String
    - getFamilienstatus gibt den Famielienstatus eines Users zurueck.
- getReligion(email : String) : String
    - getReligion gibt die Religion/en eines Users zurueck.
- getInteressen(email : String) : String
    - getInteressen gibt die Interessen eines Users zurueck.
- getGeschlecht(email : String) : String
    - getGeschlecht gibt das angegebene Geschlecht eines Users zurueck.

- getHobbies(email : String) : String
    - getHobbies gibt die Hobbies eines Users zurueck.
- getVorname(email : String) : String
    - getVorname gibt den Vornamen des Users zurueck.
- istInEinerFreundesliste(email : String, freundmail : String) : boolean
    - Gibt true zurueck, falls der eine User email in irgendeiner Form mit dem uebergebenen User freundmail in einer Freundschaftbeziehung oder auf der Freundschaftsanfragenliste einen oder des anderen ist.
- darfFreundeslisteSehen(betrachter : String, eigentuemer : String) : boolean
    - Überprüft ob der betrachter die Freundesliste des eigentuemer sehen darf.
- darfPinnwandSehen(betrachter : String, eigentuemer : String) : boolean
    - darfPinnwandSehen ueberprueft ob der betrachter die Pinnwand des eigentuemer sehen darf.
- darfProfilinfoSehen(betrachter : String, eigentuemer : String) : boolean
    - darfProfilinfo ueberprueft ob der betrachter die Profilinfo des Eigentuemer sehen darf.
- darfKontaktinfoSehen(betrachter : String, eigentuemer : String) : boolean
    - darfKontaktinfoSehen ueberprueft ob der betrachter die Kontaktinfo des eigentuemer sehen darf.
- istInGruppe(usermail : String, freundmail : String, gruppennamen : String) : boolean
    - istInGruppe ueberprueft ob der User freundmail in der Gruppe mit dem Titel gruppenname des Users usermail enthalten ist.
- istGruppe(usermail : String, freundmail : String) : boolean
    - istGruppe ueberprueft ob der User freundmail in irgendeiner Gruppe des Users usermail enthalten ist.
- gruppeErstellen(String email, Boolean[] sicht, String name) : Boolean
    - gruppeErstellen erstellt eine Gruppe und kann diese zu einer der Sichtbarkeitsliste hinzufuegen.
- gruppeLoeschen(email : String, gruppe : String) : void
    - gruppeLoeschen loescht eine Gruppe des uebergebenen Users.

- gruppeAendern(email : String, freundmail : String, gruppenname : String) : void
    - gruppeAendern aendert die Gruppenzugehoerigkeit des User freundmail der mit dem User email befreundet ist.
- profilinfoAendern(email : String, vorname : String, nachname : String, geburtsdatum : String, hobbies : String, geschlecht : String, religion : String, familienstatus : String, sexOrientierung : String, polEinstellung : String, zusatzangaben : String) : void
    - profilinfoAendern aendert die Profilinfo des uebergebenen Users.
- kontaktDatenAendern(emailID : String, adresse : String, wohnort : String, telefonnr : String, email : String) : void
    - kontaktDatenAendern aendert die Kontaktdaten des Users
- durchschnittlicheFreunde(berechtigung : String) : String
    - durchschnittlicheFreunde gibt die Anzahl der geschlossenen Freundschaften pro User zurueck.
- FreundeMonat(berechtigung : String) : String
    - FreundeMonat gibt Anzahl der geschlossenen Freundschaften pro User vom letzten Monat zurueck.
- FreundeGesamt(berechtigung : String) : String
    - FreundeGesamt gibt die Anzahl aller geschlossenen Freundschaften pro User zurückt.
- beitraegeMonat(berechtigung : String) : String
    - beitraegeMonat gibt die Anzahl der geposteten Beitraege pro User vom letzten Monat zurueck.
- durchschnittlicheBeitraege(berechtigung : String) : String
    - durchschnittlicheBeitraege gibt die Anzahl der gepostetеten Beitraege pro User.
- anzahl_userSperren(berechtigung : String) : String
    - anzahl_userSperren gibt die Anzahl der aktuellen gesperrten User zurueck.
- usersuche(email : String, nutzersuche : ArrayList<String>, id : ArrayList<String>, suchtwort : String) : void
    - usersuche ermoeglicht dem User anhand einer Eingabe einen User zu suchen.
- userSuchenHelp(stichwort : String) : ArrayList<User>
    - userSuchenHelp eigentliche Suche und sucht User anhand der Eingabe

- beitragMelden(email : String, id : int) : void
    - beitrag melden ermoeglicht das Melden eines Beitrags, der daraufhin von einem Admin ueberprueft wird.
- beitragLoeschen(email : String, id : int) : void
    - beitragLoeschen ermoeglicht das Loeschen von einem Beitrag mittels der uebergebenen id.
- beitragHinzufuegen(email : String, titel : String, inhalt : String) : void
    - beitragHinzufuegen ermoeglicht einen Beitrag hinzuzufuegen mit einem Titel und einem Inhalt.
- resaveAnwender(anwender : Anwender) : void
    - resaveAnwender loescht zunaechst das gesamte anwenderDAO und schreibt dann alles neu rein.
- getTyp(email : String) : String
    - getTyp gibt zurueck ob es sich bei der uebergebenen Email um einen Admin, Forscher oder User handelt.

1.11. Klasse Logout

Die Klasse Logout dient lediglich der Löschung der Cookies des Benutzers und leitet diesen dann zur IndexSeite weiter.

Methoden

- doGet(request : HttpServletRequest, response : HttpServletResponse) : void
    - Setzt die „TimeToLive“ des Cookies auf 0, und weißt den Nutzer dann auf die IndexSeite (/BlueCouch/) weiter.

1.12. Klasse Index

Startservlet – Standardmäßiger Einstieg des Benutzers („/BlueCouch/“ oder „/BlueCouch/Index“).

Attribute

- static aw : AnwenderManagement = new AnwenderManagement()
    - Objekt zur Verfügungstellung von Methoden des AnwenderManagements.

Methoden

- doGet(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft die Cookies des Aufrufenden Benutzers. Sollten keine (gültigen) Cookies vorhanden sein, wird dem Benutzer die „index.jsp“ angezeigt, andernfalls wird er an das jeweils für ihn gültige Servlet verwiesen (Admin → /BlueCouch/Admin, Forscher → /BlueCouch/Forscher, User → /BlueCouch/User)

1.13. Klasse Register

Die Klasse Register ermöglicht eine Registrierung in der BlueCouch mittels einer eindeutigen EmailID sowie eines Passwortes.

Attribute

- static aw : AnwenderManagement = new AnwenderManagement()
    - Objekt zur Verfügungstellung von Methoden des AnwenderManagements.

Methoden

- doGet(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft die Cookies des Aufrufenden Benutzers. Sollten keine (gültigen) Cookies vorhanden sein, wird dem Benutzer die „register.jsp“ angezeigt, andernfalls wird er an das jeweils für ihn gültige Servlet verwiesen (Admin → /BlueCouch/Admin, Forscher → /BlueCouch/Forscher, User → /BlueCouch/User)
- doPost(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft ob die, mittels eines Formulars in “register.jsp” übergebenen”, Eingaben den Vorgaben entsprechen. Wenn Nein wird „register.jsp“ samt entsprechender Fehlermeldung dem User angezeigt, Wenn Ja wird er an das jeweils für ihn gültige Servlet verwiesen (Admin → /BlueCouch/Admin, Forscher → /BlueCouch/Forscher, User → /BlueCouch/User)

1.14. Klasse FreundServlet

Die Klasse FreundServlet stellt Hauptservlet bei der Betrachtung einer UserSite eines Freundes dar. Sie überprüft welche Berechtigungen der Benutzer hat, und zeigt diesem anschließend je nachdem das entsprechende freund.jsp dar (mittels request-Attribute gesteuert)

Attribute

- static aw : AnwenderManagement = new AnwenderManagement()
    - Objekt zur Verfügungstellung von Methoden des AnwenderManagements.

Methoden

- doGet(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft zuerst die Cookies – sollten diese ungültig sein wird dem User die „forbidden.jsp“ angezeigt, andernfalls wird überprüft welche Berechtigungen der User hat, sodass ihm die jeweils veränderte „freund.jsp“ Seite angezeigt werden kann.
- doPost(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft zuerst die Cookies – sollten diese ungültig sein wird dem User die „forbidden.jsp“ angezeigt, andernfalls wird überprüft welche Berechtigungen der User hat, sodass ihm die jeweils veränderte „freund.jsp“ Seite angezeigt werden kann. Kümmert sich weiters um das Melden von Beiträgen mittels von „freund.jsp“ übergebenen Werten.

1.15. Klasse UserServlet

Klasse UserServlet stellt Hauptservlet bei der Betrachtung der eigenen Seite, so wie Pinnwand, Freunde, etc., dar. Zeigt dem Benutzer die entsprechenden Seiten(Pinnwand, Freunde, Profilinfo, Kontaktdaten, etc.) und setzt entsprechende RequestAttribute. Stellt die Schnittstelle zur Aenderung der Daten dar.

Attribute

- static aw : AnwenderManagement = new AnwenderManagement()
    - Objekt zur Verfügungstellung von Methoden des AnwenderManagements.

Methoden

- doGet(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft zuerst die Cookies auf Gültigkeit. Je nach Erfüllung dieses Kriteriums wird dem User entweder die „forbidden.jsp“ angezeigt, oder die „user.jsp“ mit der jeweils gewählten Ansicht.
- doPost(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft zuerst die Cookies auf Gültigkeit. Je nach Erfüllung dieses Kriteriums wird dem User entweder die „forbidden.jsp“ angezeigt, oder die jeweils, mittels „user.jsp“ übergebene, Aktion durchgeführt. Anschließend wird dem User wieder die (nun geänderte) user.jsp angezeigt.
- freundAdd(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Methode fügt(versendet) unter Zuhilfename des AnwenderManagement eine Freundschaftsanfrage dem Benutzer hinzu (an einen Benutzer). Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.
- freundschaftStorno(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Löscht unter Zuhilfename des AnwenderManagement eine ausgehende Freundschaftsanfrage des Benutzers. User kann eine Freundschaftsanfrage stornieren. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.
- freundschaftNein(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Löscht unter Zuhilfename des AnwenderManagement eine eingehende Freundschaftsanfrage des Benutzers. User kann eine Freundschaftsanfrage ablehnen. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.

- freundschaftJa(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Beantwortet unter Zuhilfename eine eingehende Freundschaftsanfrage und fuegt dadurch einen Freund dem Benutzer hinzu. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.
- freundDel(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Löscht unter Zuhilfename des AnwenderManagements einen Freund aus den bestehenden Freunden. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.
- freundesseite(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Methode leitet an FreundServlet weiter und setzt entsprechendes Cookie, dass die emailID des anzuzeigenden Nutzers enthaelt.
- gruppeErstellen(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Überprüft eingegebene Paramter. Bei Fehlern folgt die Weiterleitung zum.jsp mit entsprechend gesetztem Attribut zur Anzeige einer entsprechenden Fehlermeldung. Sollten Parameter richtig gesetzt sein, wird mit Hilfe des AnwenderManagements versucht eine Gruppe mit durch Parameter angegebenen Sichtbarkeitsrechten anzulegen. Sollte dies Fehlschlagen erneut auf user.jsp samt Fehlermeldung verwiesen. Andernfalls erfolgt die Weiterleitung an die GET Methode des UserServlets
- gruppeDel(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Löscht unter Zuhilfename des AnwenderManagement eine Gruppe aus den bestehenden Gruppen. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.
- gruppeAendern(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Ändert unter Zuhilfename des AnwenderManagement eine Gruppe aus den bestehenden Gruppen. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.
- neuerBeitrag(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void
    - Fügt unter Zuhilfename des AnwenderManagement einen neuen Beitrag zu den bestehenden Beitraegen hinzu. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p. Im Fehlerfall wird zu user.jsp weitergeleitet wo entsprechende Fehlermeldung ausgegeben wird.

- profilinfoaendern(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void

    - Ändert unter Zuhilfename des AnwenderManagement die bestehende Profilinfo des Benutzers. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p. Im Fehlerfall wird zu user.jsp weitergeleitet wo entsprechende Fehlermeldung ausgegeben wird.

- kontaktdatenaendern(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void

    - Ändert unter Zuhilfename des AnwenderManagement die bestehenden Kontaktdaten des Benutzers. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.

- beitragloeschen(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void

    - Löscht unter Zuhilfename des AnwenderManagement einen Beitrag aus den bestehenden Beitraegen des Benutzers. Anschließend erfolgt Weiterleitung zu User Seite mit gesetztem Parameter p.

- usersuche(cookie: Cookie, request : HttpServletRequest, response: HttpServletResponse) : void

    - Sucht unter Zuhilfename des AnwenderManagement einen User aus den bestehenden Usern.

1.16. Klasse AdminServlet

Die Klasse AdminServlet überprüft ob eine entsprechende Berechtigung mittels Cookies vorhanden ist (= ob Benutzer als User vom Typ Admin eingloggt ist) und stellt mittels Methoden des AnwenderManagements eine entsprechende Möglichkeit dar, durch Anweisungen mittels „admin.jsp“ Änderungen vorzunehmen (= gemeldete Beiträge freigeben/löschen sowie User deren Beiträge gemeldet wurden zu sperren)

Attribute

- static aw : AnwenderManagement = new AnwenderManagement()
    - Objekt zur Verfügungstellung von Methoden des AnwenderManagements.

Methoden

- doGet(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft die Cookies – bei ungültigen Cookies wird die „forbidden.jsp“ angezeigt, andernfalls die „admin.jsp“.
- doPost(request : HttpServletRequest, response : HttpServletResponse) : void
    - Empfängt Daten welche mittels “admin.jsp” übergeben wurden und führt entsprechende Aktionen mithilfe Methoden der Klasse AnwenderManagement aus. Hierzu zählen: Beitrag löschen, Beitrag freigeben sowie User sperren. Anschließend wird wieder die nun geänderte „admin.jsp“ angezeigt.

1.17. Klasse ForscherServlet

Die Klasse ForscherServlet überprüft ob eine entsprechende Berechtigung mittels Cookies vorhanden ist (= ob Benutzer als User vom Typ Forscher eingloggt ist) und stellt mittels Methoden des AnwenderManagements eine entsprechende Möglichkeit dar, durch Anweisungen mittels „forscher.jsp“ Auswertungen vorzunehmen und auszugeben.

Attribute

- static aw : AnwenderManagement = new AnwenderManagement()
    - Objekt zur Verfügungstellung von Methoden des AnwenderManagements.

Methoden

- doGet(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft die Cookies – bei ungültigen Cookies wird die „forbidden.jsp“ angezeigt, andernfalls die „forscher.jsp“.
- doPost(request : HttpServletRequest, response : HttpServletResponse) : void
    - Empfängt Daten, welche mittels “forscher.jsp” übergeben werden und weißt entsprechende request Attribute hinzu (mittels Werte welche mithilfe Methoden der Klasse AnwenderManagement erhalten werden). Anschließend wird dem Benutzer die (nun geänderte) „forscher.jsp“ angezeigt.

1.18. Klasse Login

Die Klasse Login ermöglicht ein Einloggen in der BlueCouch mittels einer eindeutigen EmailID sowie eines Passwortes.

Attribute

- static aw : AnwenderManagement = new AnwenderManagement()
    - Objekt zur Verfügungstellung von Methoden des AnwenderManagements.

Methoden

- doGet(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft die Cookies des Aufrufenden Benutzers. Sollten keine (gültigen) Cookies vorhanden sein, wird dem Benutzer die „login.jsp“ angezeigt, andernfalls wird er an das jeweils für ihn gültige Servlet verwiesen (Admin → /BlueCouch/Admin, Forscher → /BlueCouch/Forscher, User → /BlueCouch/User)
- doPost(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft ob die, mittels eines Formulars in “register.jsp” übergebenen”, Eingaben den Vorgaben entsprechen. Wenn Nein wird „register.jsp“ samt entsprechender Fehlermeldung dem User angezeigt, Wenn Ja wird er an das jeweils für ihn gültige Servlet verwiesen (Admin → /BlueCouch/Admin, Forscher → /BlueCouch/Forscher, User → /BlueCouch/User)

1.19. Klasse Upload

Die Klasse Upload ist für den Upload einer (Bild-)Datei zuständig. Sie überprüft ob eine uploadbare Datei vorliegt, und ob diese vom Typ „/Image“ ist. Wenn Ja, wird es entsprechend geuploaded und mittels Methoden des AnwenderManagements dem jeweiligen User zugeordnet.

Attribute

- static aw : AnwenderManagement = new AnwenderManagement()
    - Objekt zur Verfügungstellung von Methoden des AnwenderManagements.
- static id : int = 0
    - stellt ID der UploadDateien dar, dient der Möglichkeit Dateien hochzuladen, welche den gleichen Namen tragen.

Methoden

- doPost(request : HttpServletRequest, response : HttpServletResponse) : void
    - Überprüft die Cookies des Benutzers. Sollten diese ungültig sein, wird diesem lediglich die „forbidden.jsp“ angezeigt. Andernfalls wird die Datei auf Gültigkeit überprüft und hochgeladen.

1.20. Klasse AnwenderDAO

Die Klasse AnwenderDAO dient ausschließlich der persistenten Speicherung, sowie dem Auslesen der persistent gespeicherten Daten. Sie übernimmt keine logische Überprüfung – die muss bereits vor Aufruf vom AnwenderManagement stattgefunden haben.

Attribute

- final DATEIPFAD : String
    - Der Dateipfad, an dessen Stelle die entsprechende Datei zur speicherung (und natürlich zum auslesen) der Daten zu finden ist.

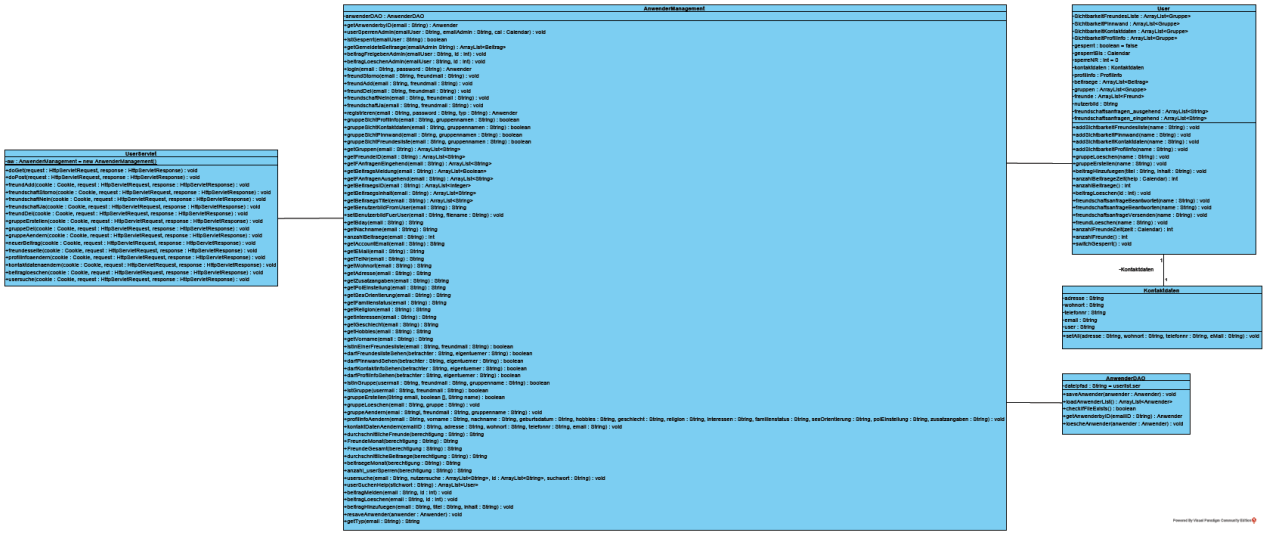
Methoden

- loadAnwenderList() : ArrayList<Anwender>
    - Die Klasse AnwenderDAO liest die gesamt persistent gepspeicherte Anwenderliste aus, und retourniert diese.
- getAnwenderbyID(id : String) : Anwender
    - Die Klasse AnwenderDAO sucht innerhalb der persistent gespeicherten Anwenderliste nach einem bestimmten User und gibt diesen ggfs. zurück. Andernfalls wird null zurückgegeben.
- saveAnwender(user : Anwender) : boolean
    - Die Klasse AnwenderDAO fügt einen Anwender zur persistent gespeicherten Anwenderliste hinzu.
- getAnwenderlistbyString(text : String) : ArrayList<Anwender>
    - Die Klasse AnwenderDAO sucht innerhalb der persistent gespeicherten Anwenderliste, nach allen Usern die einem bestimmten String gleichen und gibt diese zurück.
- loescheAnwender(anwender : Anwender) : void
    - Die Klasse AnwenderDAO entfernt einen Anwender aus der persistent gespeicherten Anwenderliste.
- checkIfFileExists() : boolean
    - Überprüft ob der Pfad (= die Datei) in der Instanzvariable DATEIPFAD bereits existiert.

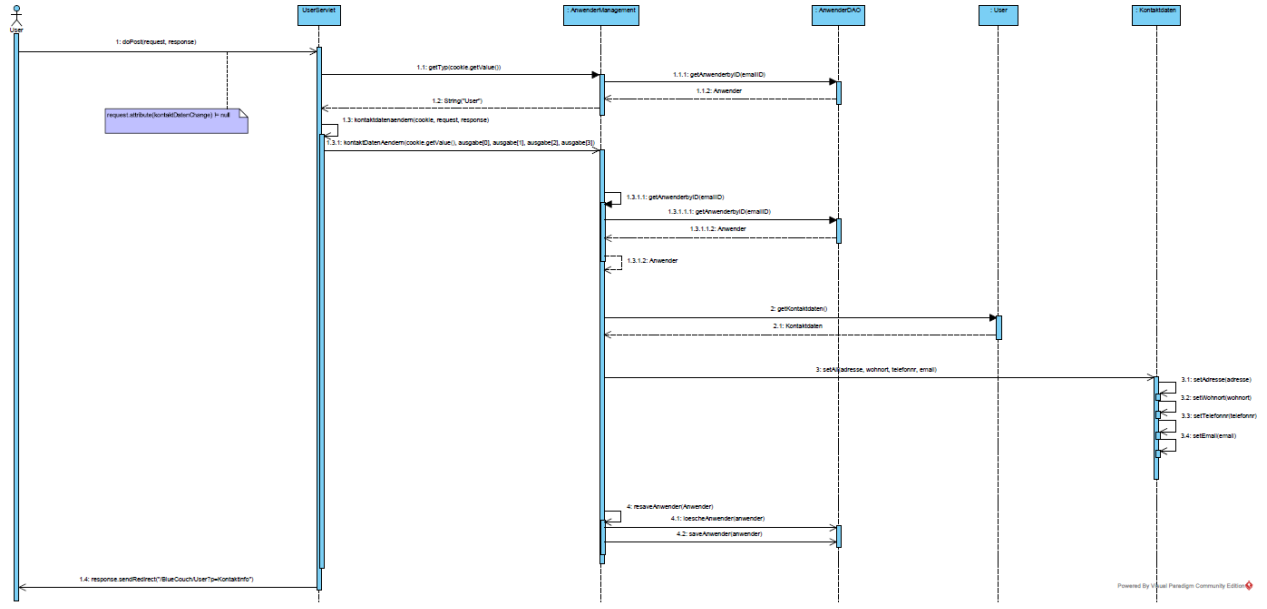
2 Use Case Realization Design

2.1 UseCase 1: Kontaktinformationen ändern

Klassendiagramm – Ausschnitt



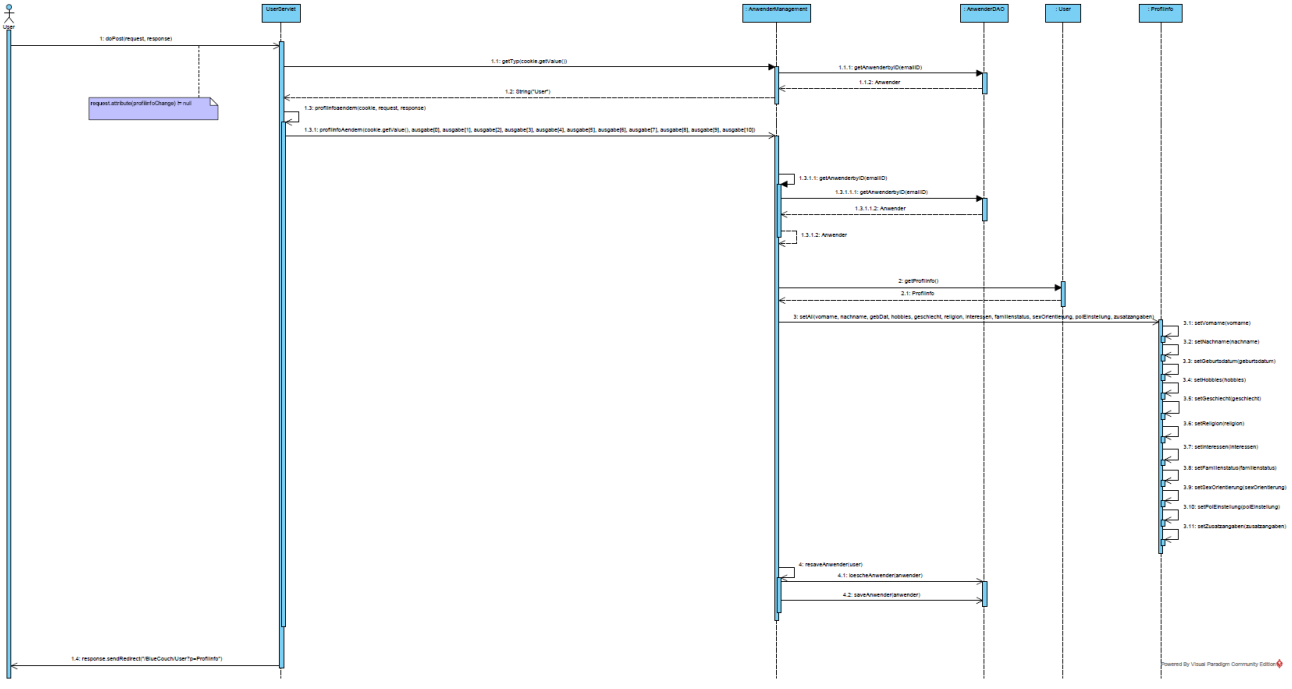
Sequenzdiagramm – Ablauf



2.2 UseCase 2: Persönliche Information ändern

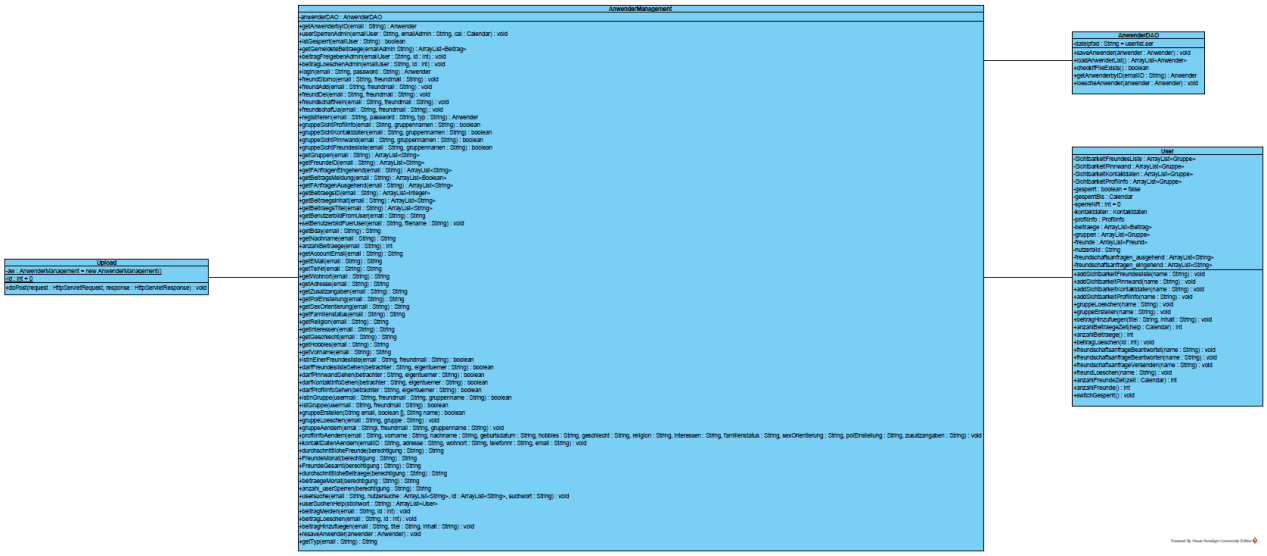
Klassendiagramm – Ausschnitt

Sequenzdiagramm – Ablauf



2.3 UseCase 3: Benutzerbild ändern

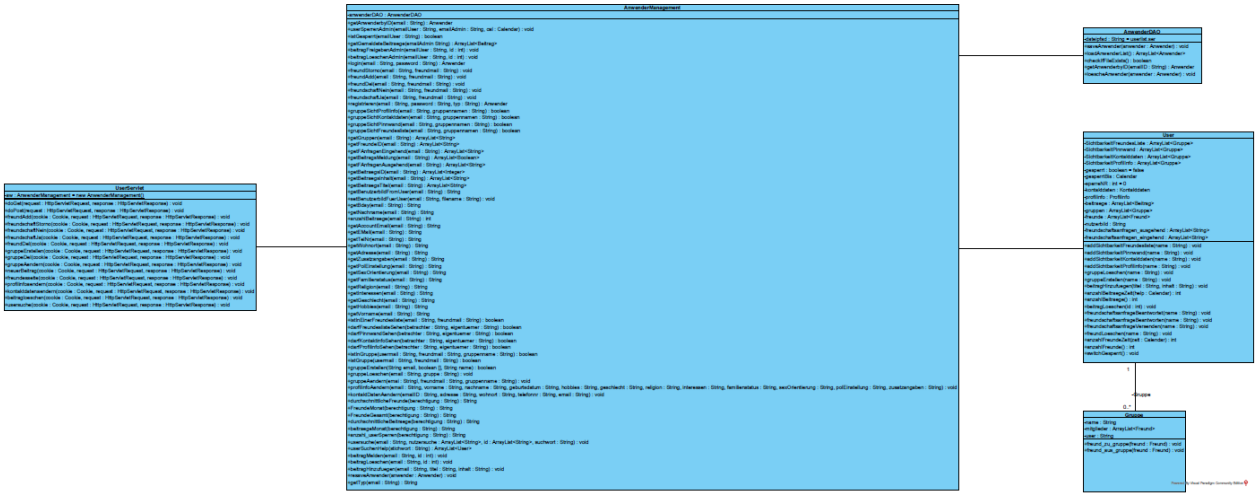
Klassendiagramm – Ausschnitt



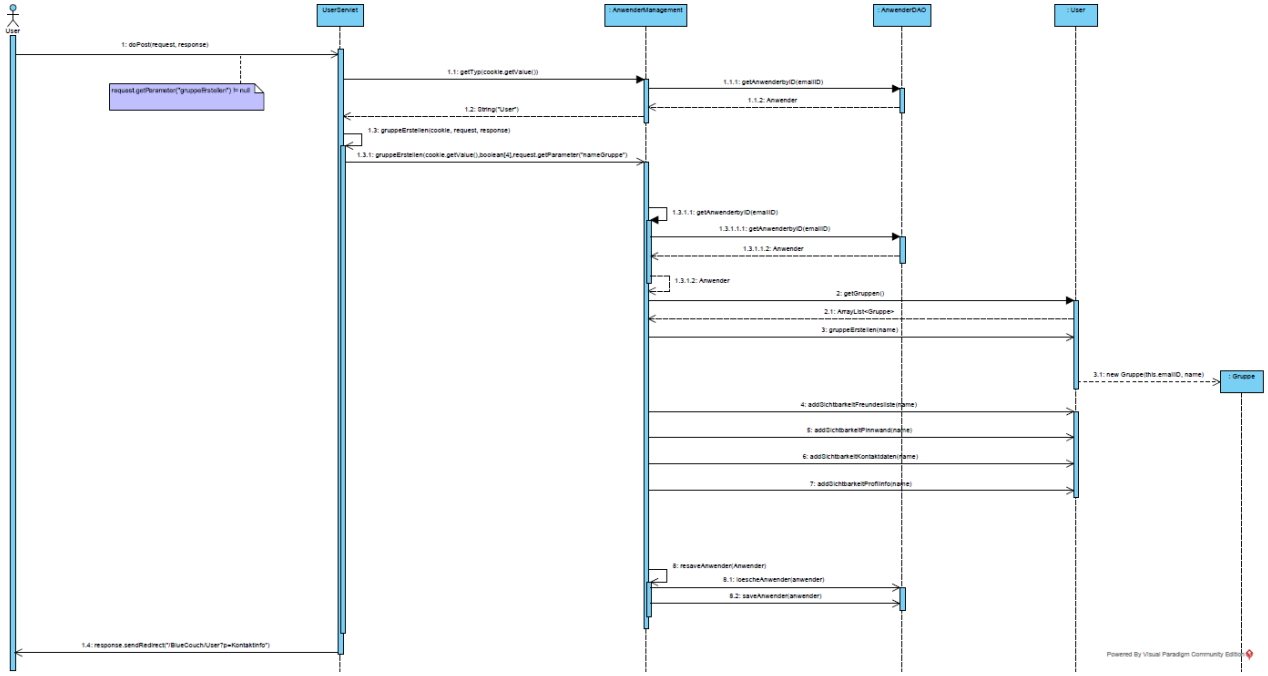
Sequenzdiagramm – Ablauf

2.4 UseCase 4: Gruppe samt Sichtbarkeitsrechte erstellen

Klassendiagramm – Ausschnitt

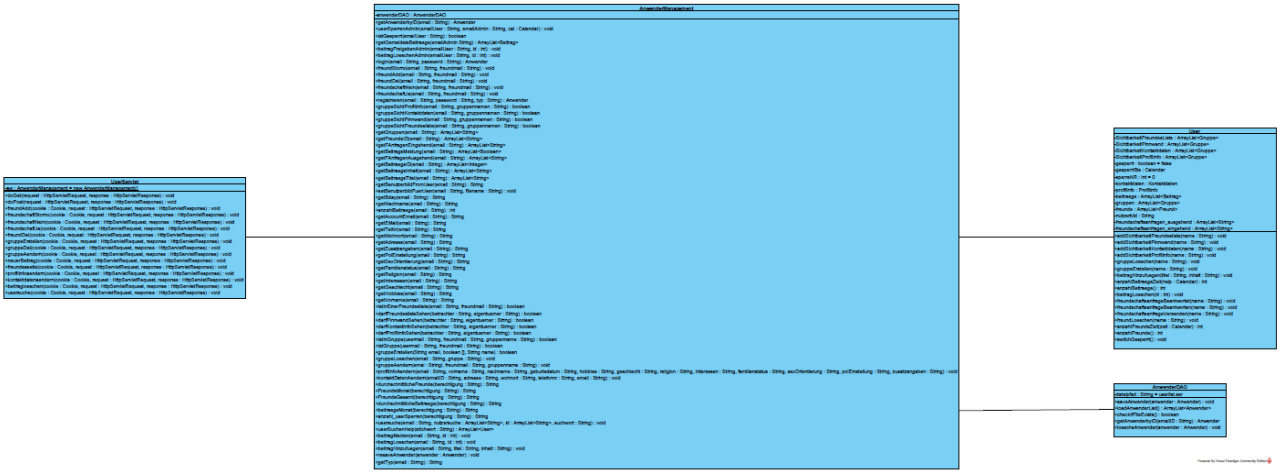


Sequenzdiagramm – Ablauf

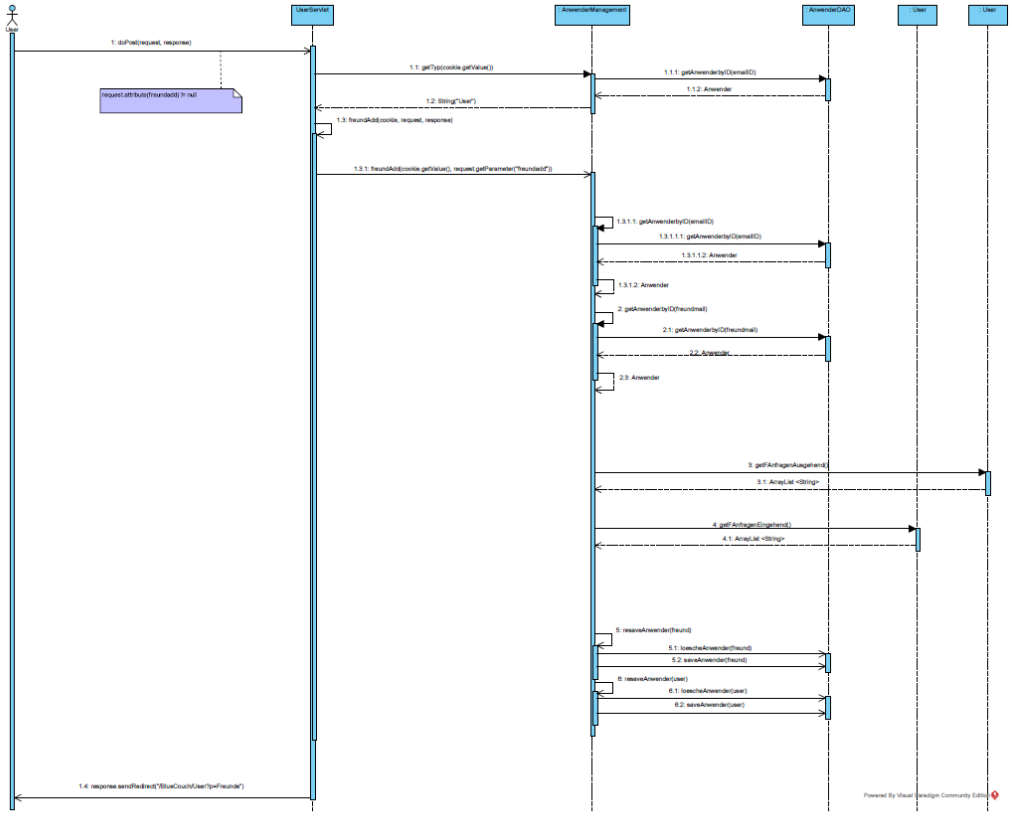


2.5 UseCase 5: Freund einladen

Klassendiagramm – Ausschnitt

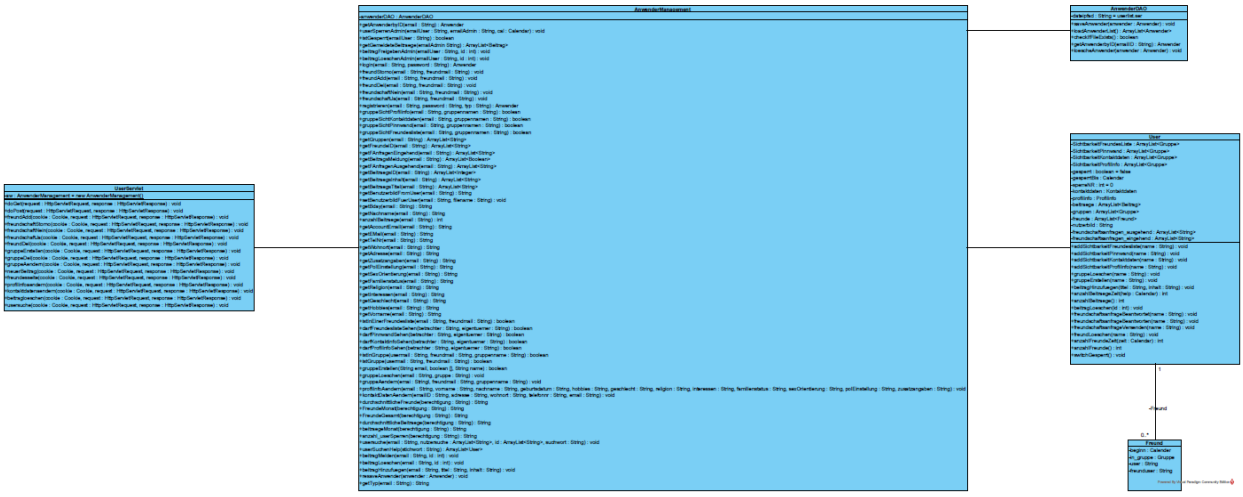


Sequenzdiagramm – Ablauf

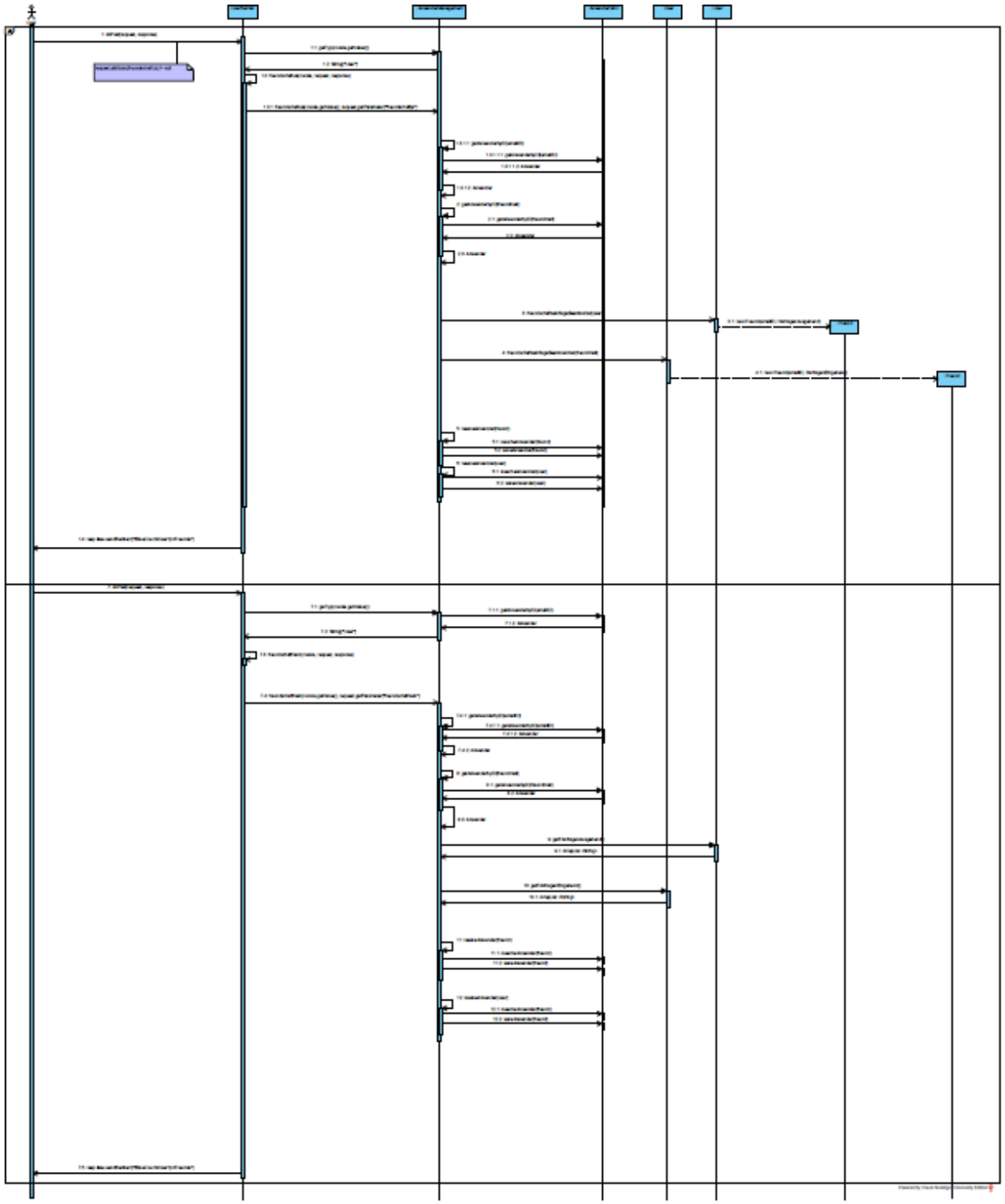


2.6 UseCase 6: auf Freundschaftseinladung reagieren

Klassendiagramm – Ausschnitt



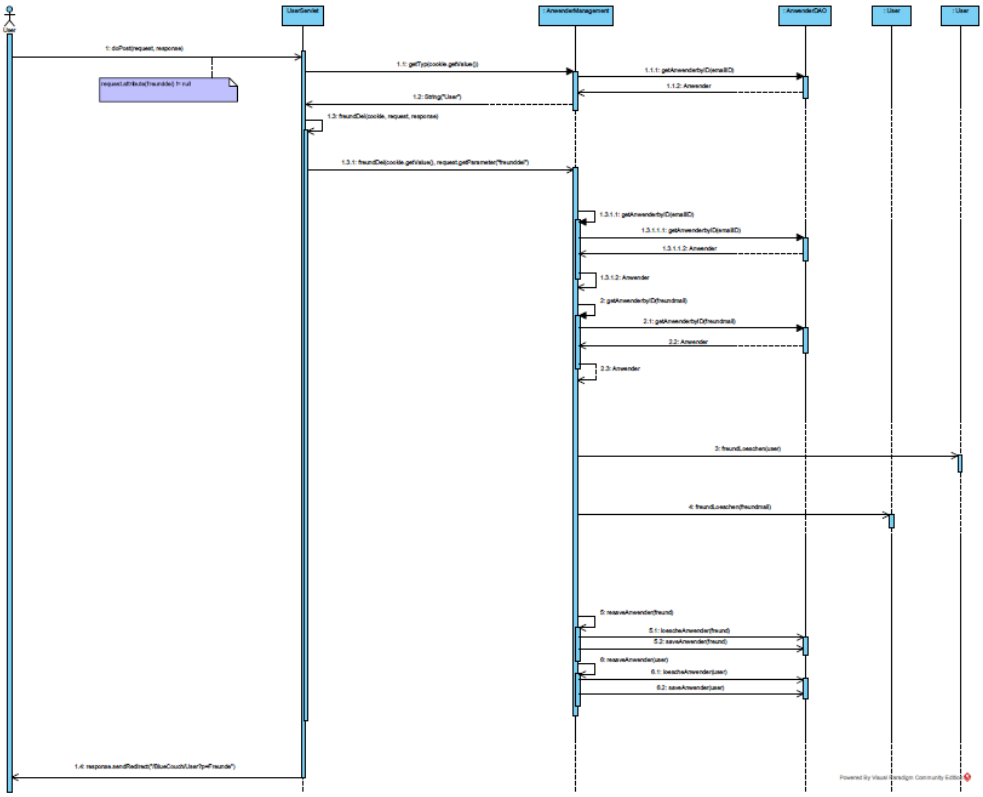
Sequenzdiagramm – Ablauf



2.7 UseCase 7: Freund aus Freundesliste löschen

Klassendiagramm – Ausschnitt

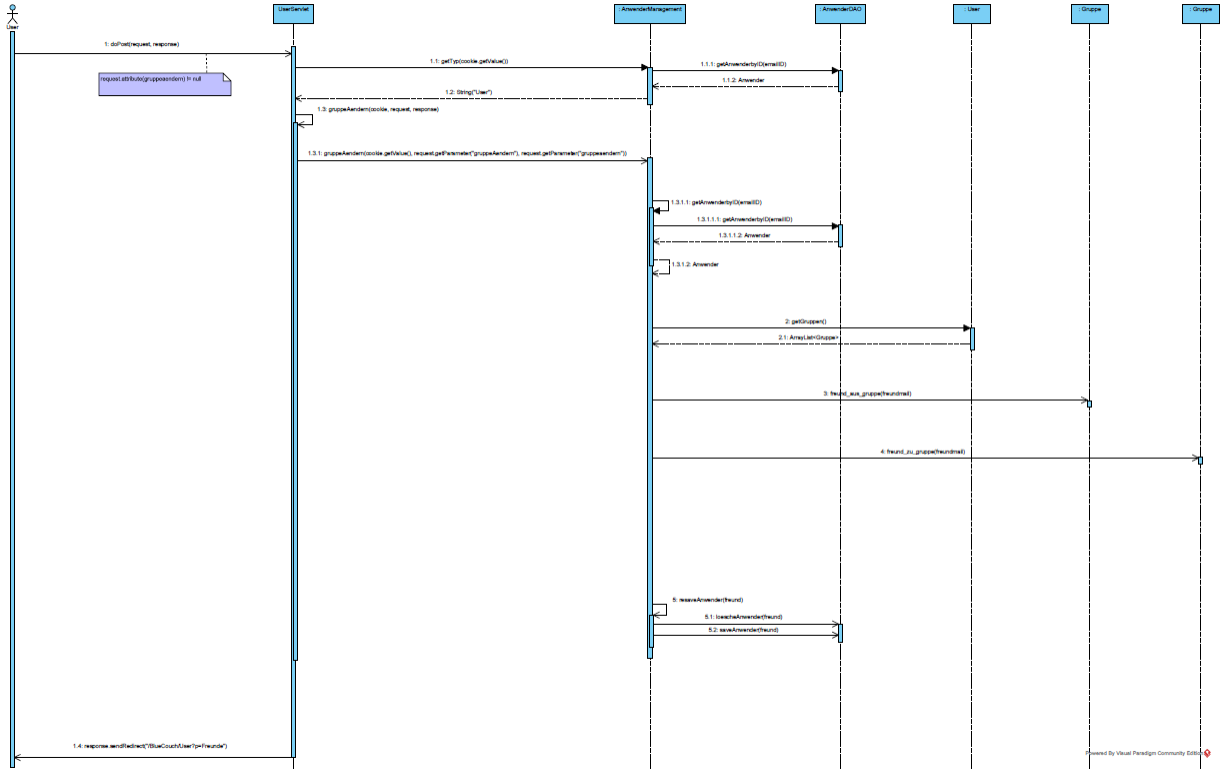
Sequenzdiagramm – Ablauf



2.8 UseCase 8: Freunde zu Gruppe zuordnen

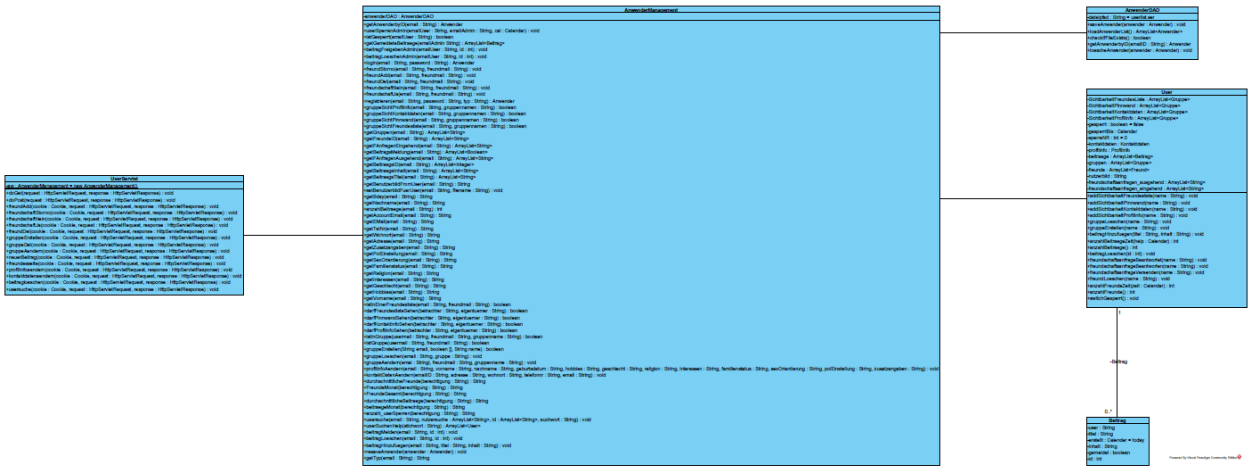
Klassendiagramm – Ausschnitt

Sequenzdiagramm – Ablauf

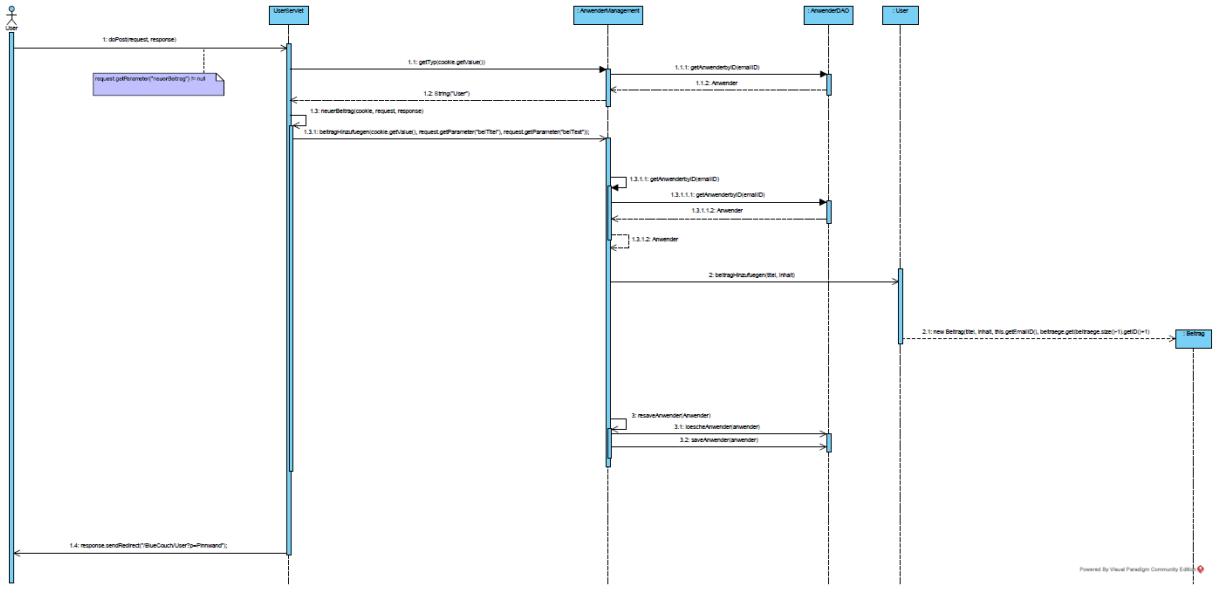


2.9 UseCase 9: Beitrag erstellen

Klassendiagramm – Ausschnitt

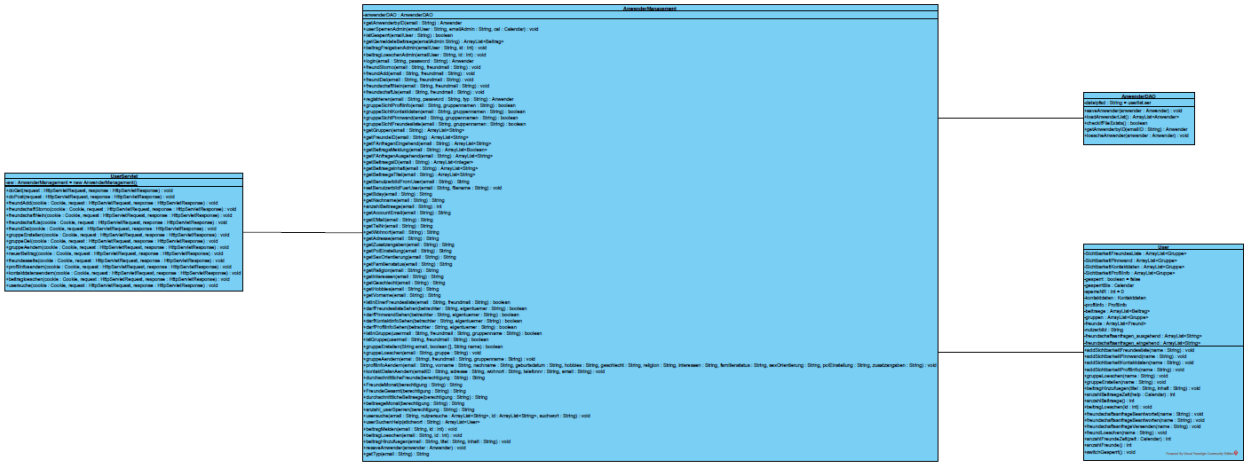


Sequenzdiagramm – Ablauf

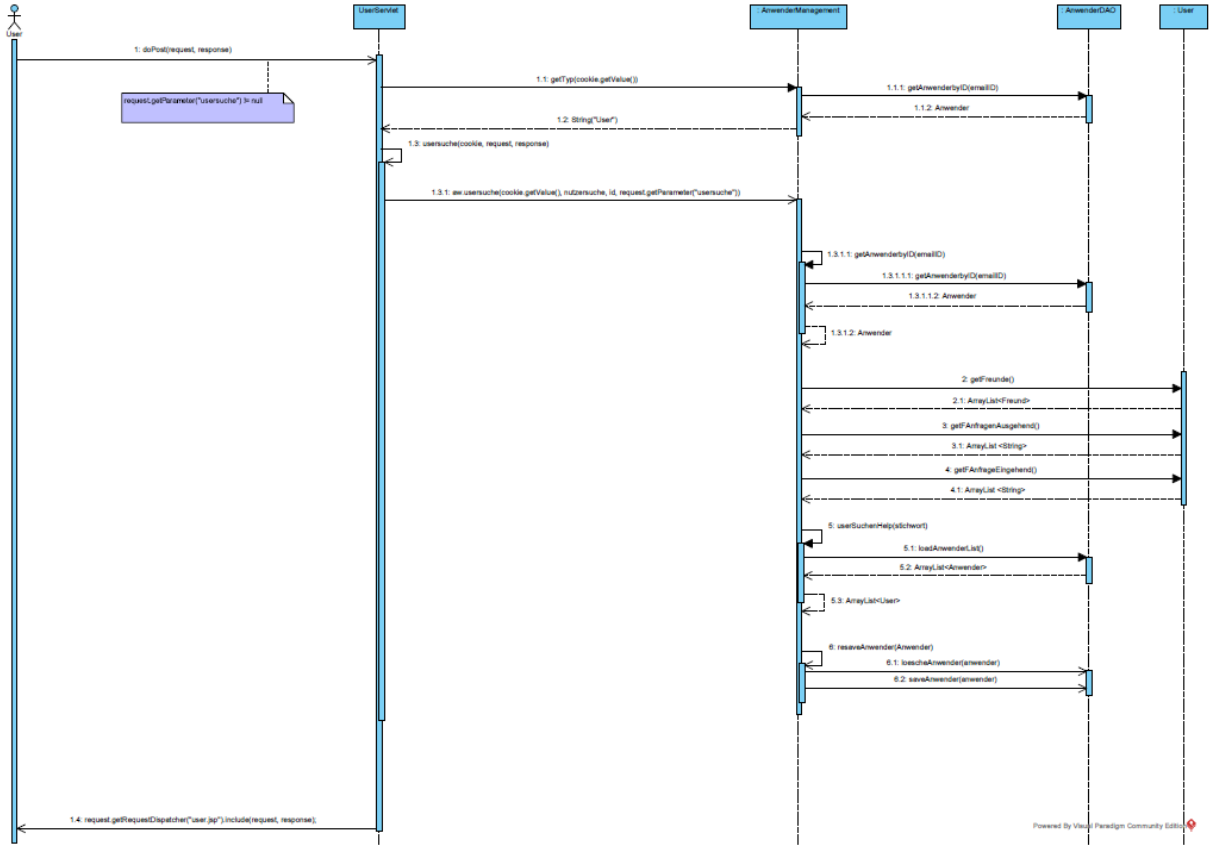


2.10 UseCase 10: User suchen

Klassendiagramm – Ausschnitt



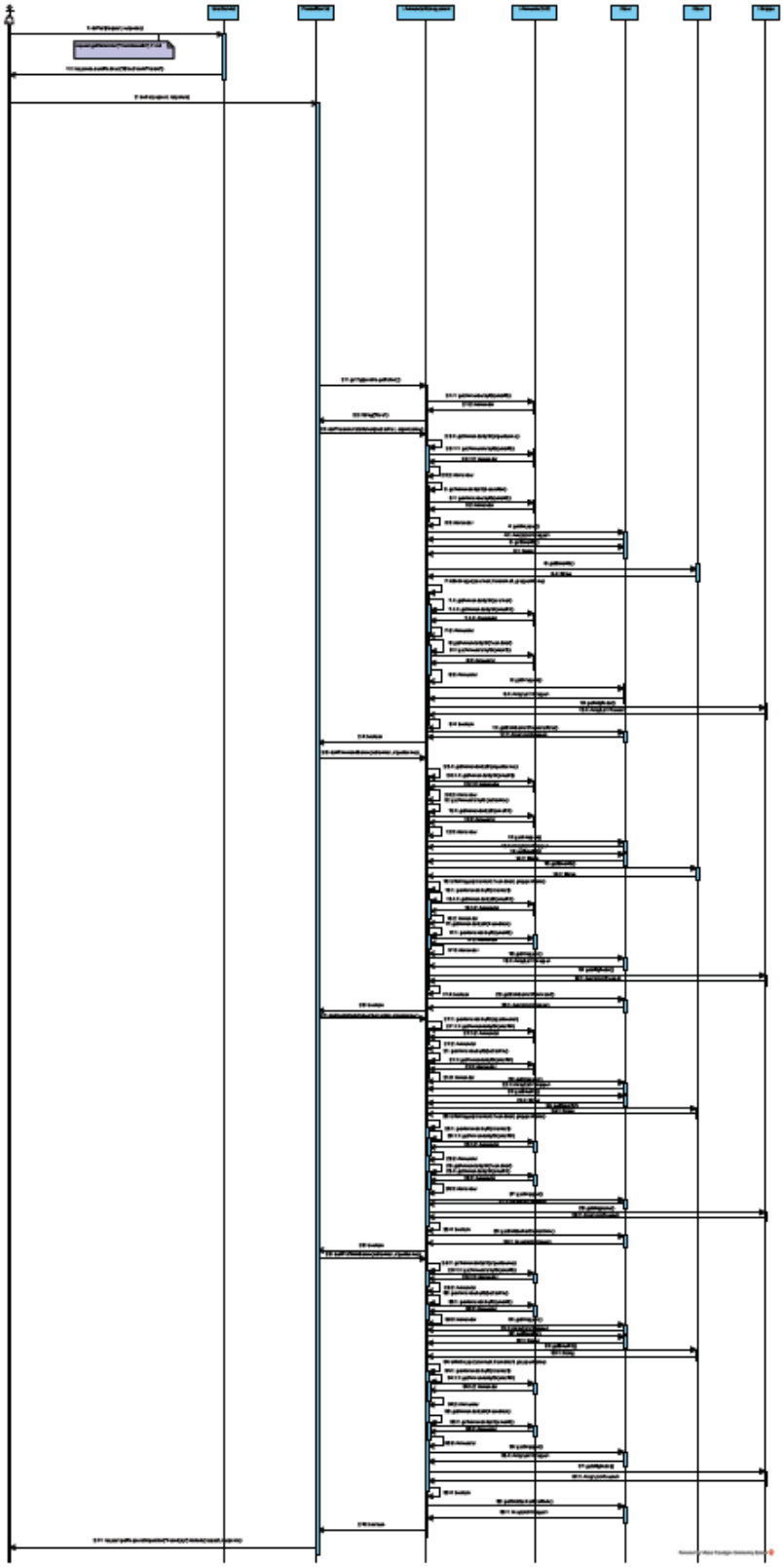
Sequenzdiagramm – Ablauf



2.11 UseCase 11: UserSite eines Freundes ansehen

Klassendiagramm – Ausschnitt

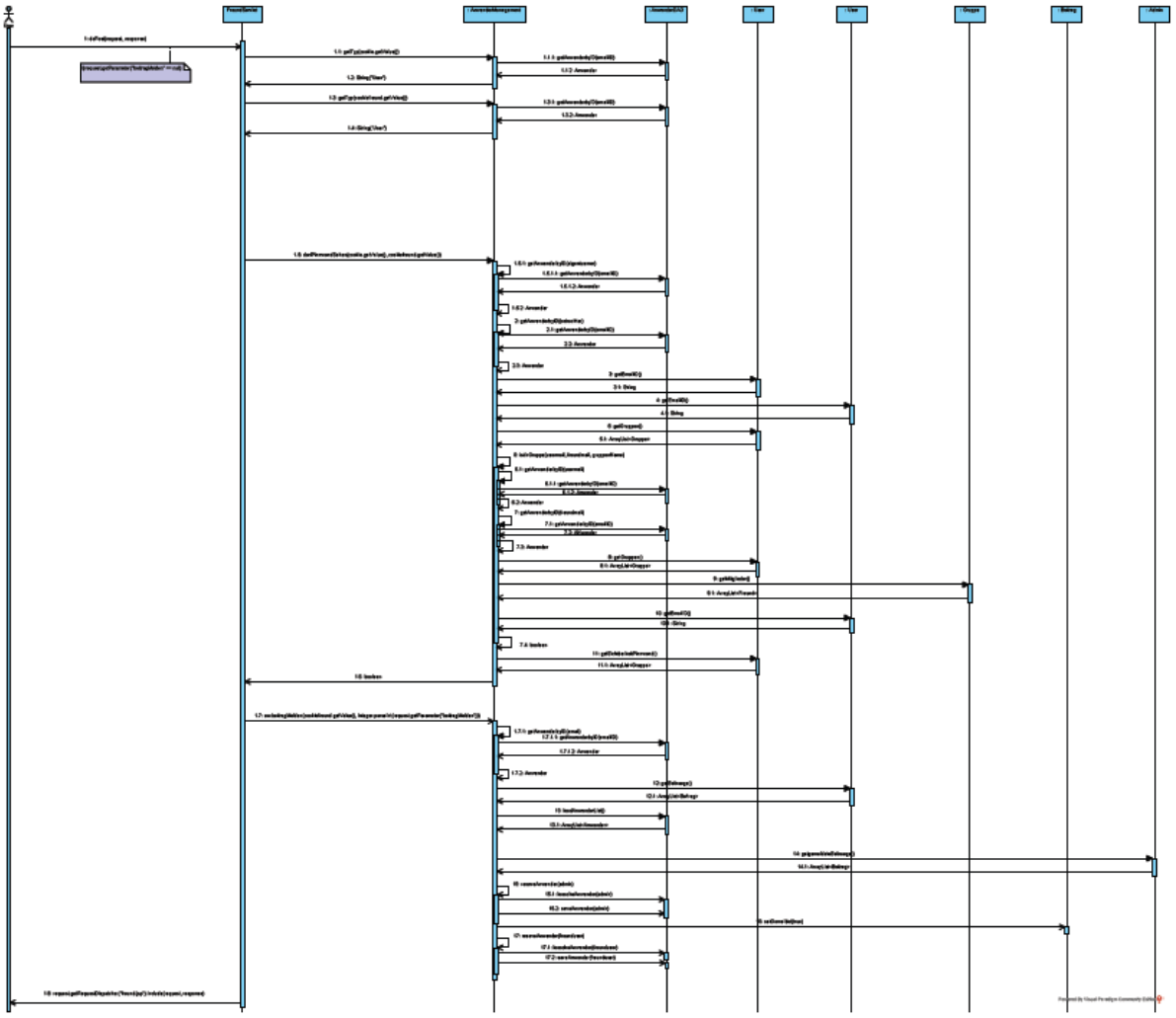
Sequenzdiagramm – Ablauf



2.12 UseCase 12: Beitrag eines Freundes melden

Klassendiagramm – Ausschnitt

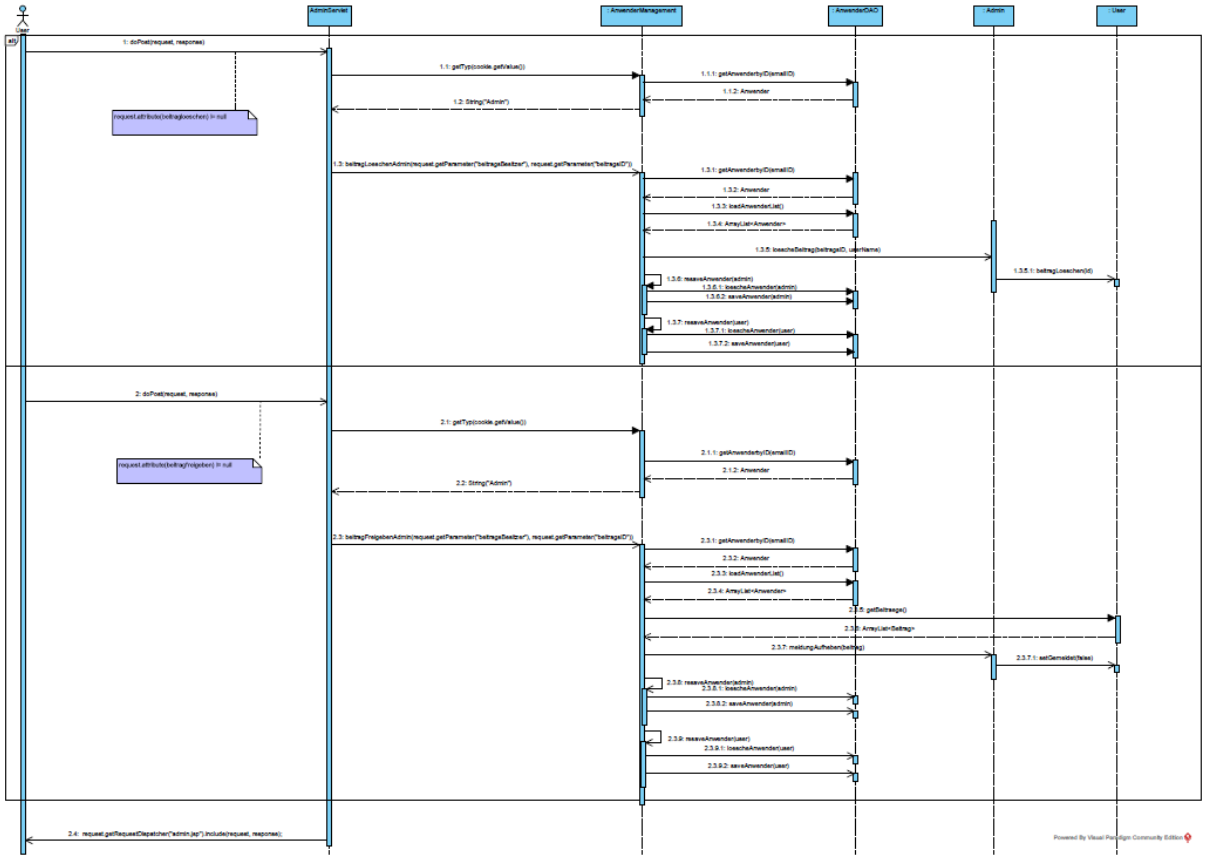
Sequenzdiagramm – Ablauf



2.13 UseCase 13: gemeldeten Beitrag prüfen

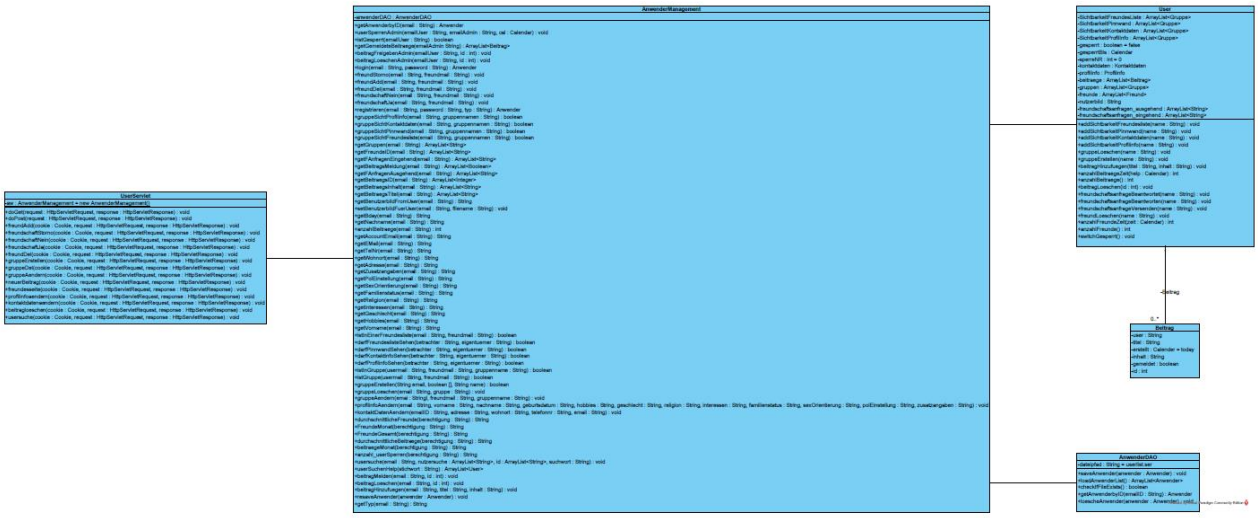
Klassendiagramm – Ausschnitt

Sequenzdiagramm – Ablauf

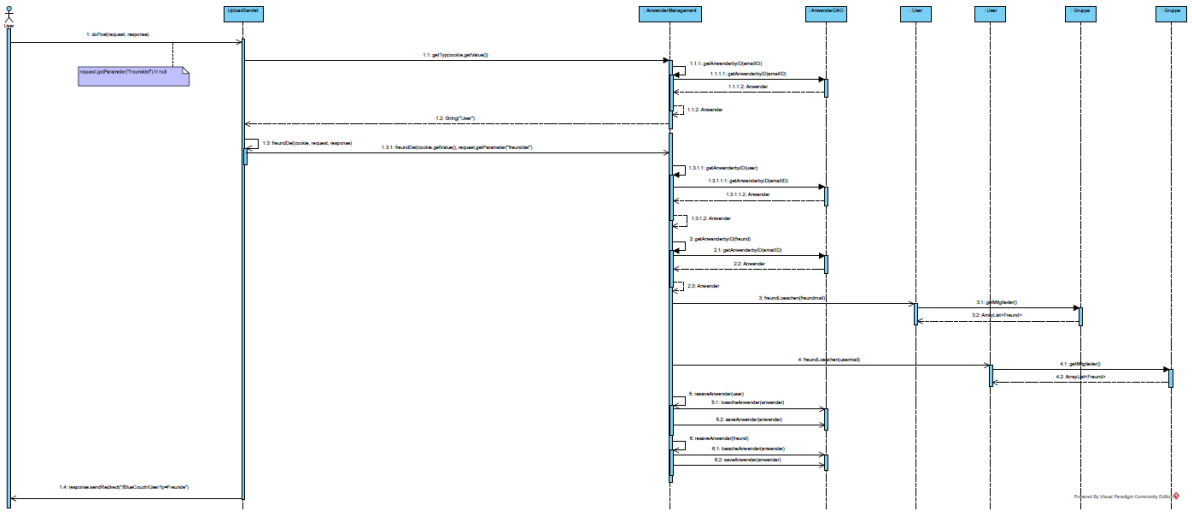


2.14 UseCase 14: Beitrag von Pinnwand löschen

Klassendiagramm – Ausschnitt

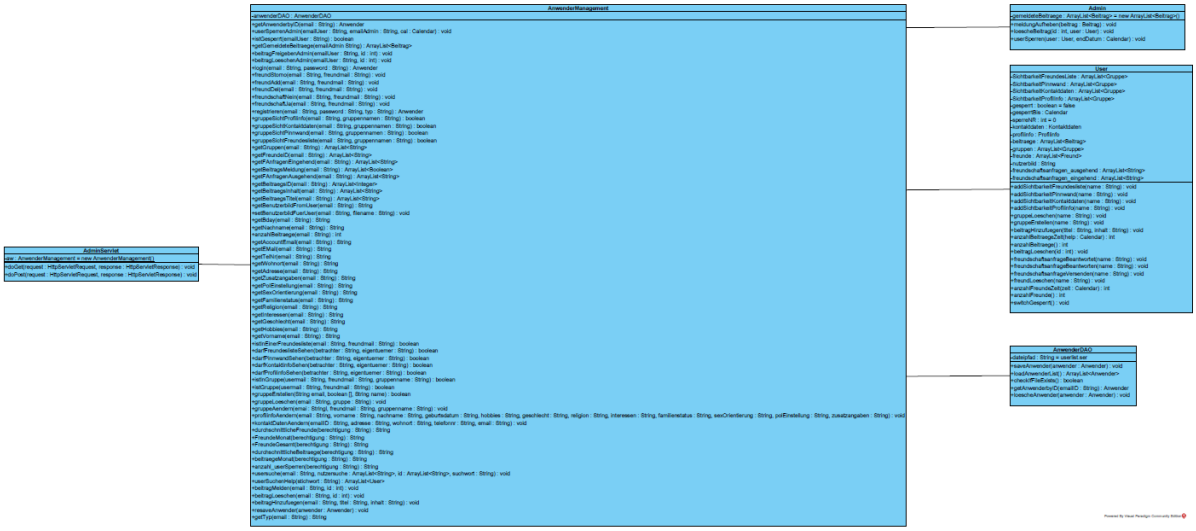


Sequenzdiagramm – Ablauf

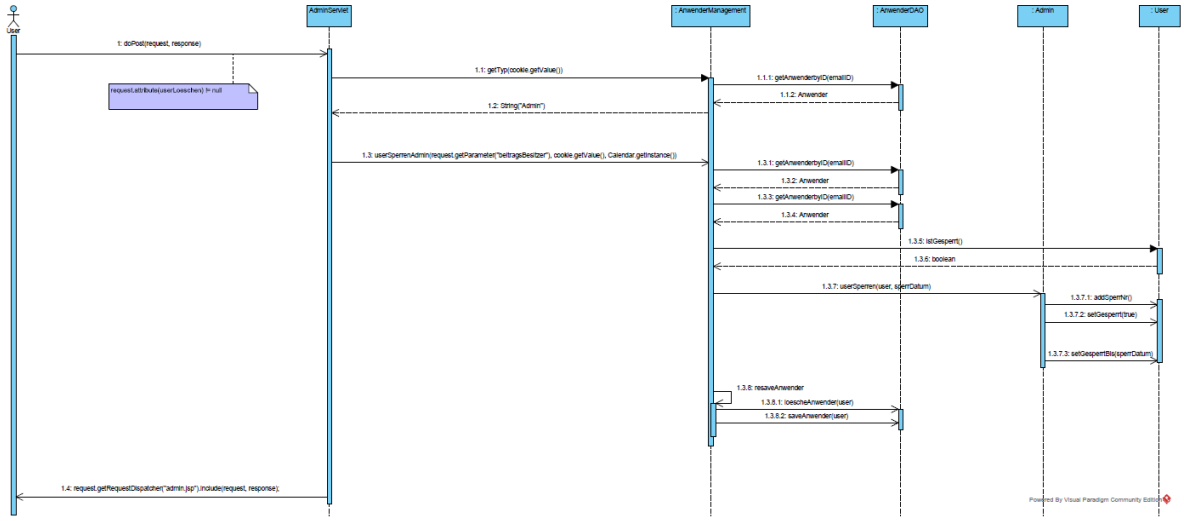


2.15 UseCase 15: User sperren

Klassendiagramm – Ausschnitt

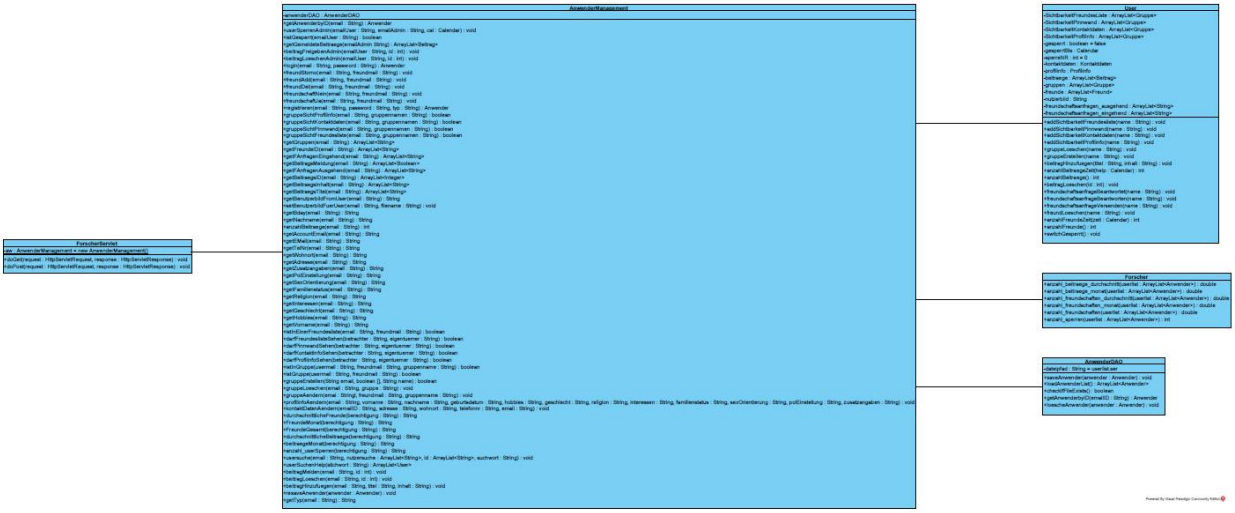


Sequenzdiagramm – Ablauf

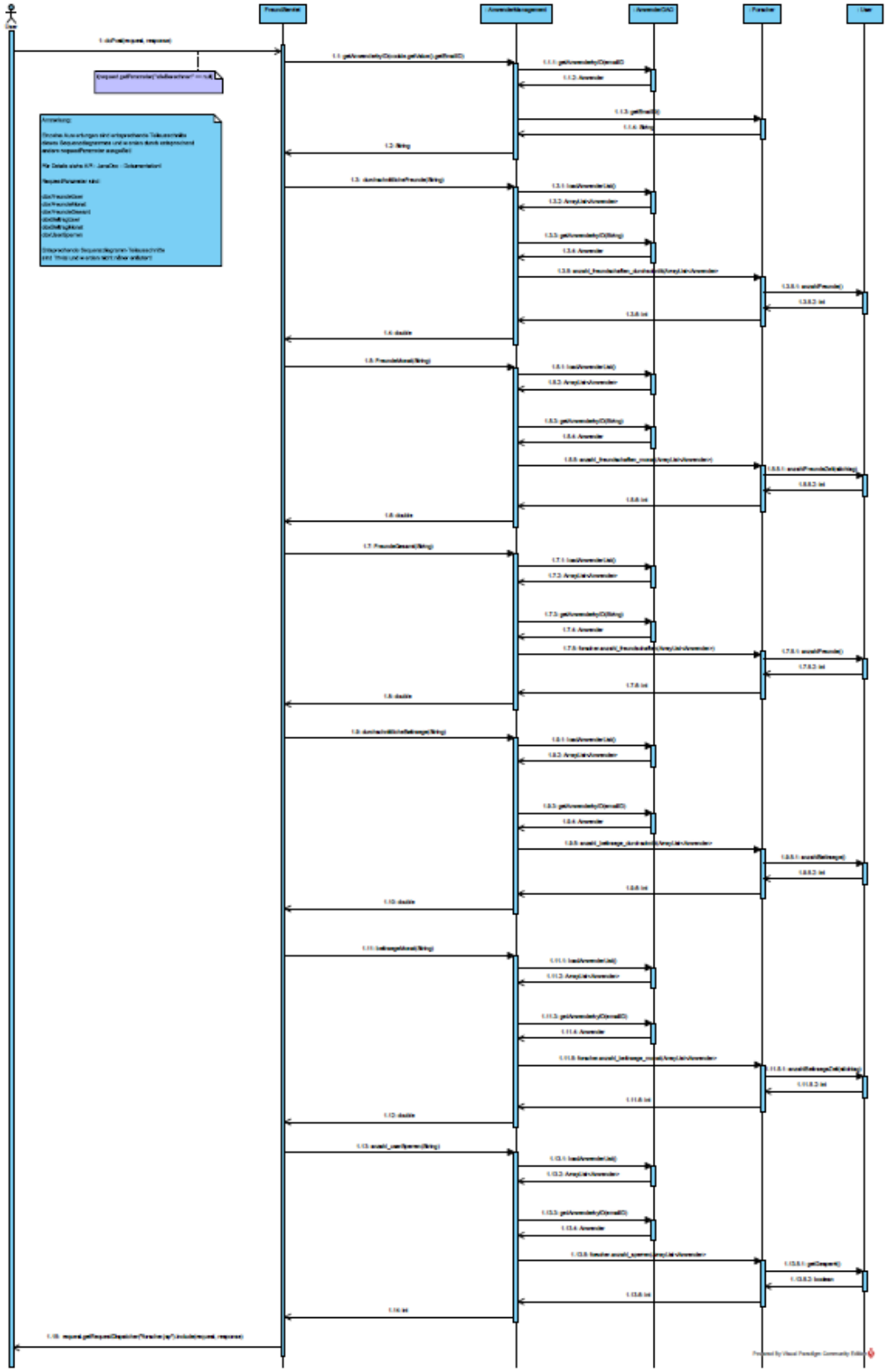


2.16 Use Case 16: Statistik auswerten

Klassendiagramm – Ausschnitt

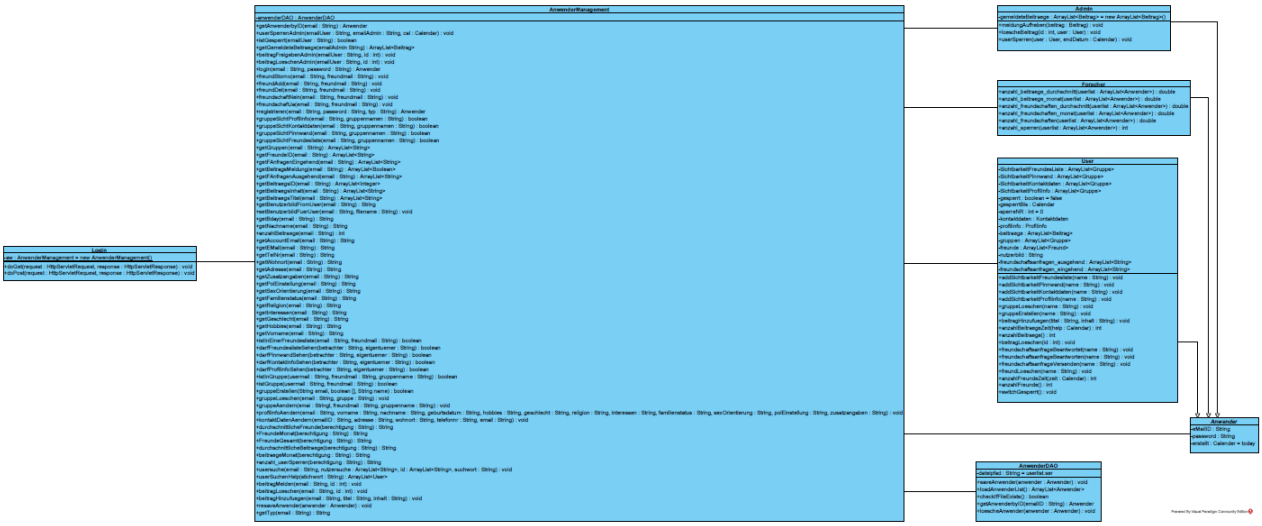


Sequenzdiagramm – Ablauf

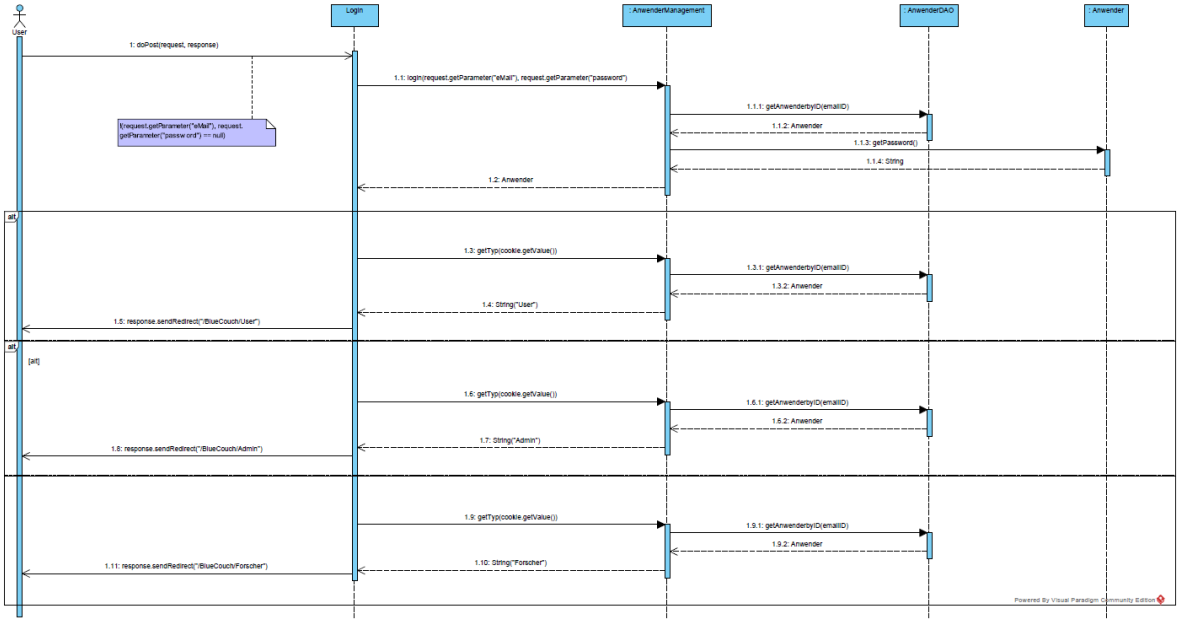


2.17 UseCase 17: Login

Klassendiagramm – Ausschnitt

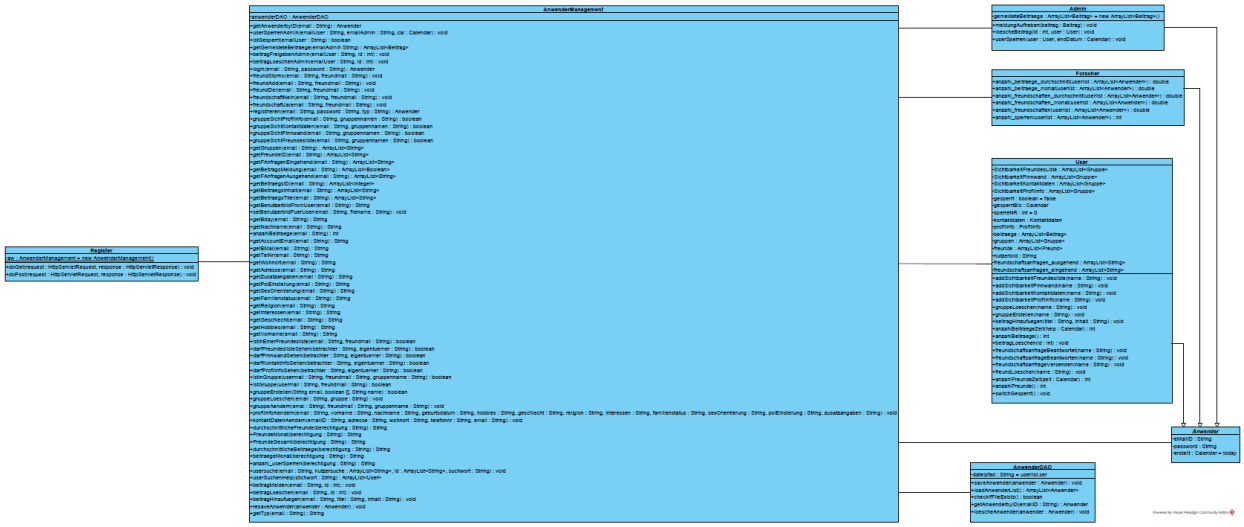


Sequenzdiagramm – Ablauf

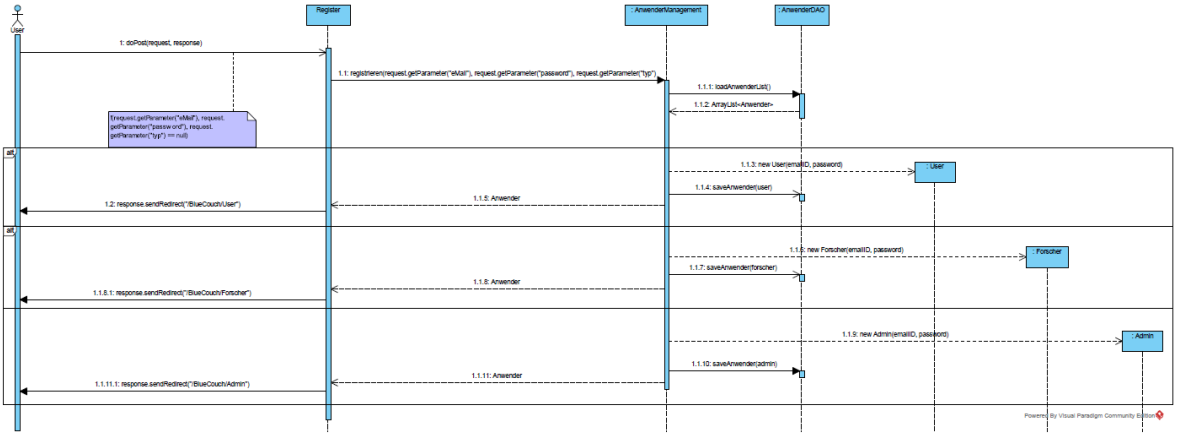


2.18 Use Case 18: Registrieren

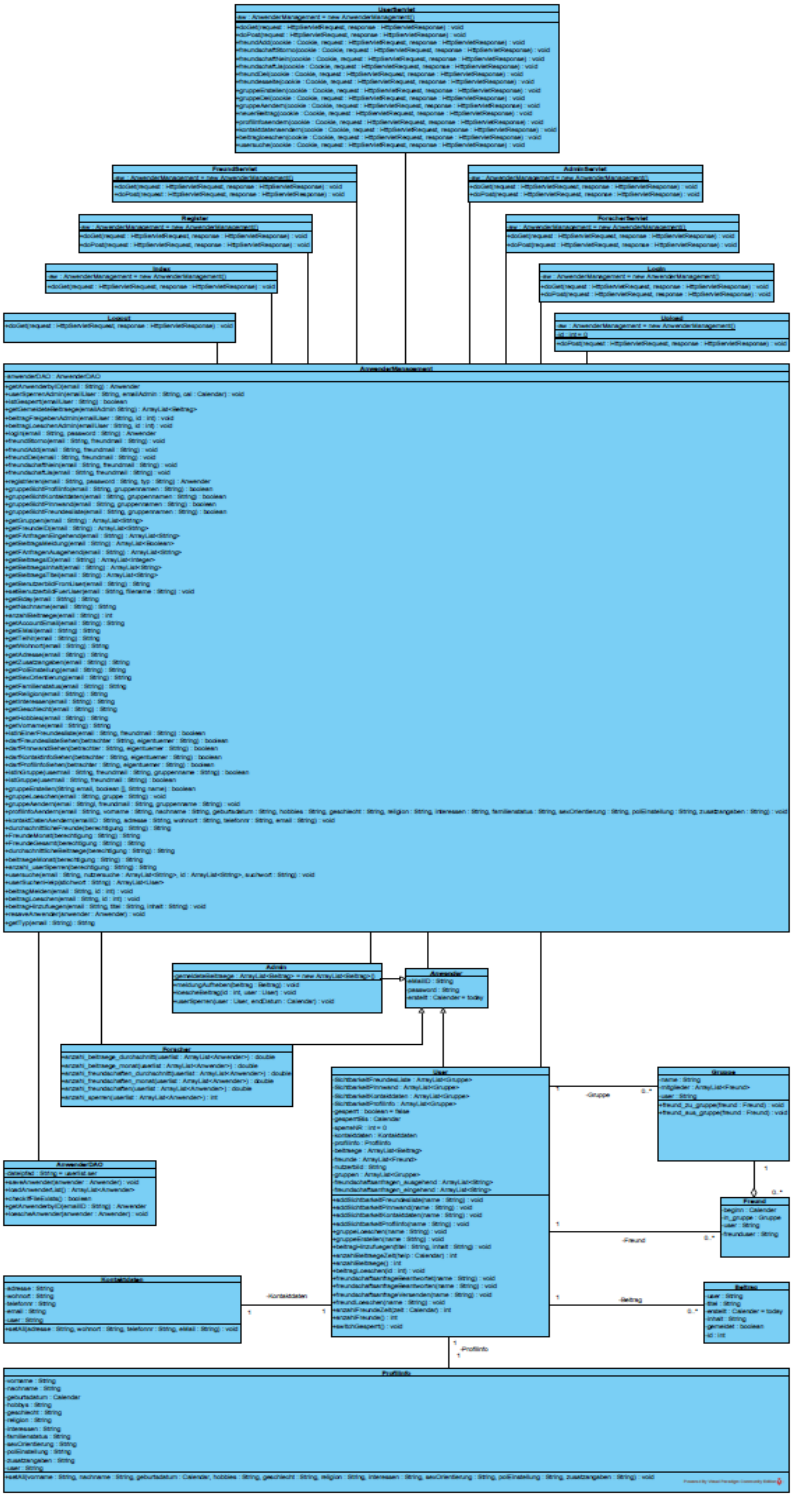
Klassendiagramm – Ausschnitt



Sequenzdiagramm – Ablauf

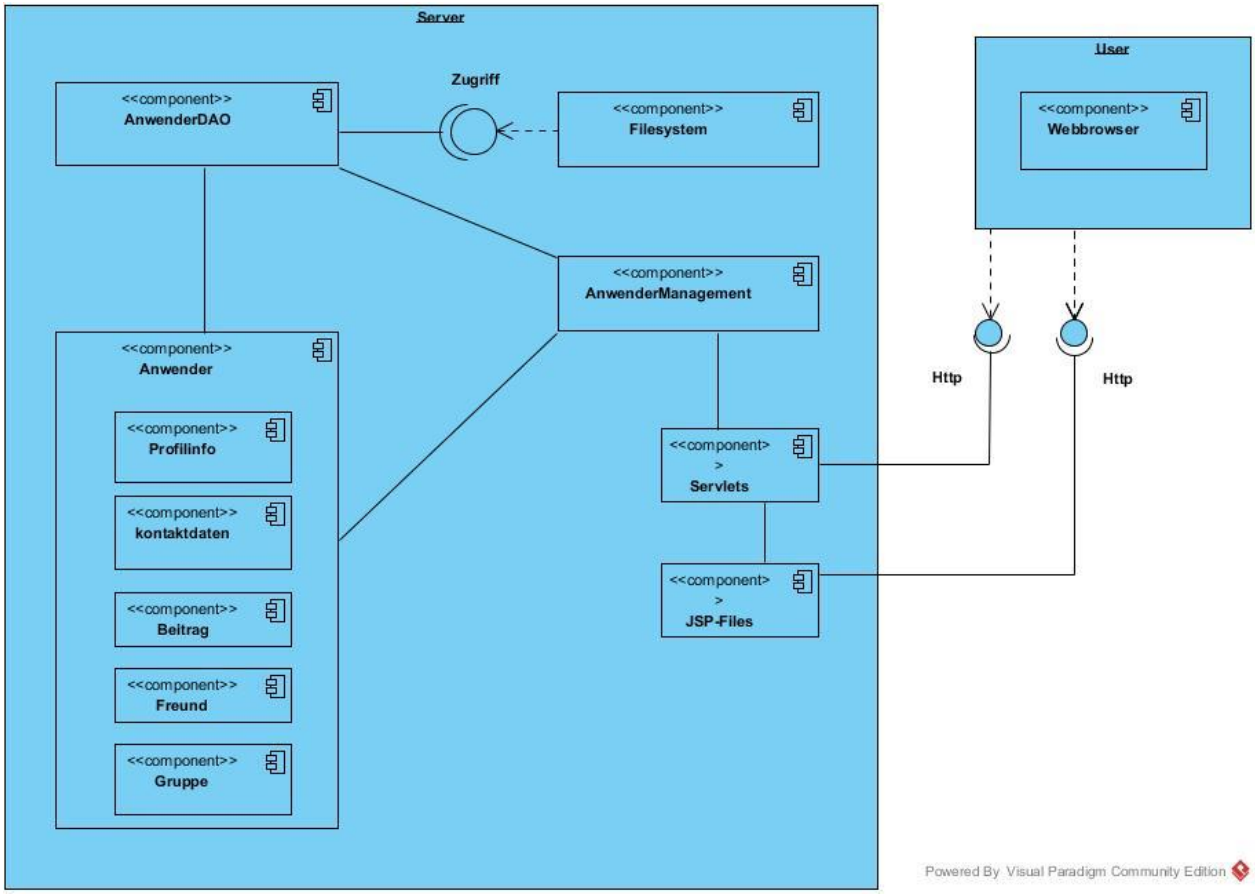


3 Übersichtsklassendiagramm



4 Architekturbeschreibung

4.1 Komponentendiagramm:



4.2 Deployment diagramm