GSA Online Plus

|  |
| --- |
| Modell-Beschreibung  Inhalt  [1 Systemeigenschaften 3](#_Toc479356249)  [2 Entwicklungsmethodik 4](#_Toc479356250)  [3 Prozesse 4](#_Toc479356251)  [3.1 Initialisierung 4](#_Toc479356252)  [3.2 Systemzugang 6](#_Toc479356253)  [3.3 Registrierung 7](#_Toc479356254)  [3.4 Hauptprozess 9](#_Toc479356255)  [3.5 Schreibaufgabe bearbeiten/Patientenstatus 10](#_Toc479356256)  [3.6 Erinnerungen/Timer 11](#_Toc479356257)  [3.7 Profil bearbeiten 12](#_Toc479356258)  [3.8 Tagebuch führen 13](#_Toc479356259)  [3.9 Ersten Tagebucheintrag schreiben 15](#_Toc479356260)  [3.10 Weitere Tagebucheinträge schreiben 16](#_Toc479356261)  [3.11 Tagebücher kommentieren 17](#_Toc479356262)  [4 View (Formularentwürfe/Wireframes) 18](#_Toc479356263)  [4.1 Admin-Formular 18](#_Toc479356264)  [4.2 Anmeldeformulare 19](#_Toc479356265)  [4.3 Tagebucheintrag 21](#_Toc479356266)  [4.4 Tagebuch 23](#_Toc479356267)  [4.5 Wochenübersicht 24](#_Toc479356268)  [4.6 Patientenprofil 24](#_Toc479356269)  [4.7 Patientenliste 25](#_Toc479356270)  [4.8 Maileditor 27](#_Toc479356271)  [5 Controller 28](#_Toc479356272)  [5.1 AdminController 28](#_Toc479356273)  [5.2 AuxController 29](#_Toc479356274)  [5.3 ContactController 29](#_Toc479356275)  [5.4 Days 30](#_Toc479356276)  [5.5 DiaryController 31](#_Toc479356277)  [5.6 DiaryEntry 32](#_Toc479356278)  [5.7 GateController 32](#_Toc479356279)  [5.8 PatientController 35](#_Toc479356280)  [5.9 PatientListController 36](#_Toc479356281)  [5.10 Timer 36](#_Toc479356282)  [5.10.1 Timer diagram 37](#_Toc479356283)  [6 Model (Daten) 39](#_Toc479356284)  [6.1 Constants / Basic Classes diagram 39](#_Toc479356285)  [6.2 Fragen nach dem Befinden 39](#_Toc479356286)  [6.3 Main Model diagram 39](#_Toc479356287)  [6.4 Answer 40](#_Toc479356288)  [6.5 AnswersPHQ4 40](#_Toc479356289)  [6.6 AnswersReply 41](#_Toc479356290)  [6.7 Assignment 41](#_Toc479356291)  [6.8 AssignmentStatus 41](#_Toc479356292)  [6.9 AssignmentTemplate 42](#_Toc479356293)  [6.10 Code 43](#_Toc479356294)  [6.11 Comment 43](#_Toc479356295)  [6.12 CommentReply 43](#_Toc479356296)  [6.13 DayOfWeek 43](#_Toc479356297)  [6.14 Patient 44](#_Toc479356298)  [6.15 PatientStatus 46](#_Toc479356299)  [6.16 Question 47](#_Toc479356300)  [6.17 QuestionsIDs 47](#_Toc479356301)  [6.18 Situation 47](#_Toc479356302)  [6.19 Survey 48](#_Toc479356303)  [6.20 Survey 48](#_Toc479356304)  [6.21 Task 48](#_Toc479356305)  [6.22 TaskTemplate 49](#_Toc479356306)  [6.23 User 49](#_Toc479356307) |
| 7.4.2017 18:50:17 |
|  |

Inhalt

# Systemeigenschaften

Diese Dokumente enthalten Informationen, die parallel zur Entwicklung des GSA-Online+-Systems gesammelt wurden.

Der finale Quelltext der Anwendung kann in Einzelheiten abweichen. Eine HTML-Dokumentation der aktuellen Quellen steht – ebenso wie dieses Dokument – im Verzeichnis doc des Repositories zur Verfügung.

**Systemeigenschaften**

Das System ermöglicht es den Teilnehmern, in einem strukturierten Prozess über 13 Wochen anonym Schreibaufgaben zu bearbeiten, die von Betreuern (Therapeuten) gestellt und kommentiert werden. Das System wurde für die Online-betreuung der Teilnehmer in kritischen Lebenssituationen entwickelt. Darüber hinaus kann es als Mittel zur anonymen Kommunikation zwischen Teilnehmern und Betreuern eingesetzt werden.

Wichtige Funktionen sind

* Video-Informationen über Ziel und Vorgehen der Intervention
* Anonyme Registrierung mit Code und Email-Adresse
* Möglichkeit der Änderung des Passworts
* Rollen: Patienten (Teilnehmer), Therapeut (Betreue), Administrator
* Erinnerung der Teilnehmer per Mail an neue Aufgaben, bald fällige und überfällige Aufgaben sowie Information über neue Kommentare der Betreuer. Erinnerungen werden automatisch verschickt, der Administrator wird über das erfolgreiche Versenden der Erinnerungen informiert.
* https-gesicherte Übermittlung von Aufgaben, Antworten der Teilnehmer und Kommentare der Betreuer
* Übermittlung von Links zu externen Fragebögen bei Beginn und Ende der Intervention
* Teilnehmer wählen einen Wochentag als Schreibtag, sie können diesen Tag (einmal) ändern
* Zusätzlich zu den Antworten auf die Schreibaufgaben schätzen die Teilnehmer wöchentlich ihren Zustand ein.
* Teilnehmer können die Kommentare der Betreuer bewerten. Diese Bewertungen sind nur für Administratoren sichtbar
* Betreuer können das Startdatum der Intervention über das Entlassungsdatum festlegen
* Betruer können Teilnehmern zugeordnet werden, jedoch kann jeder Betreuer jeden Patienten betruen und dess Daten, mit Ausnahme der eMail-Adresse, einsehen
* Beteruer können Notizen zu Teilnehmern anlegen
* Betreuer können Muster für Schreibaufgaben anlegen. Die initialen Schreibaufgaben der ersten Woche sind standardisiert.
* Zu jedem Tag von Sonntag bis Donnerstag kann die Anzahl der Teilnehmer begrenzt werden, die diesen Tag als Schreibtag wählen können. Betreuer können diese Anzahl verändern.
* Die Teilnehmerliste zeigt den Status jedes Patienten einschließlich des Standes der Bearbeitung der aktuellen Schreibaufgabe an. Sie kann gefiltert und soriert werden.
* Zu jedem Patienten kann eine Übersicht (Wochenrückblick) aller bisherigen Schreibaufgaben, Antworten und Kommentare erzeugt werden. Diese Übersicht enthält eine grafische Darstellung der Entwicklung der Selbsteinschätzung des Teilnehmers.
* Zugriffsstatistiken werden mit piwik gesammelt. der Teilnehmer kann der Sammlung seiner Daten widersprechen.
* Im Nicht-Produktivbetrieb kann das System zu Test- und Schulungszwecken mit Dummy-Daten gefüllt werden. In diesem Modus kann das aktuelle Datum über die unter /test erreichbare Testoberfläche manipuliert werden.

**Technik**

Das System ist in PHP 7 mit folgenden Hilfsmitteln implementiert.

* MySQL
* Apache 2
* Nodejs
* Laravel 5
* Bootstrap
* Ajax

Für die Verwaltung der Quellen wiurde [Gitlab](https://gitlab.uni-koblenz.de/iwm/gsa-online-plus) eingesetzt. Das System wird über die php-Dateien im Ordner *config* konfiguriert.

# Entwicklungsmethodik

Das Modell ermöglicht eine Aufteilung der Arbeiten wie folgt.

1. Server-Entwicklung: Zustände mit Aktionen nur  auf dem Server sind mit fetter grüner Schrift markiert, die übrigen Zustände warten auf Eingaben des Benutzers über die damit verbundenen Dialoge. Des weiteren beschreiben die Transitionen zwischen den Zuständen Aktionen auf dem Server. Aus dem Modell geht hervor, welche Servermethoden benötigt werden (s.u.), jedoch nicht wie sie zu implementieren sind. Das [Datemodell](file:///\\iwmfs\dahn\Aktuelles\Zwerenz\GSA\$package:\%7b8FC7031E-84F7-4c70-9636-37DA17421AAC%7d) gibt Hinweise zur Strukturierung der Daten. Buttons senden [Signale](file:///\\iwmfs\dahn\Aktuelles\Zwerenz\GSA\$package:\%7b4878951C-BBE7-407a-B029-E481AB0CC167%7d) die auf dem Server von den Methoden (in EA Operationen) der Klasse Anmelder verarbeitet werden.
2. Client-Entwicklung: Ausgangspunkt für die Entwicklung der benötigten Webseiten sind die [Win32-Wireframes](file:///\\iwmfs\dahn\Aktuelles\Zwerenz\GSA\$package:\%7b3AFAC7AF-2FC0-4012-9499-C1A69615051F%7d). Einige dieser Seiten benötigen eine interne Funktionalität (z.B. zur Prüfung von Nutzereingaben) die im Modell dokumentiert aber nicht modelliert ist.Eine weitere Grundlage für die Cliententwicklung sind diebei den Win32-Dialogen verlinkten Balsamiq-Wireframes.
3. Webdesign: Ebenfalls auf der Grundlage der Win32-Dialoge und der Balsamiq-Wireframes sollten Seitendesigns entworfen werden die genau die Kontrollelemente der Win32-Dialoge präsentieren.
4. Testentwicklung: Die Szenarien der Anmelde-Use-Cases sollten so überarbeitet werden dass sie den state machines entsprechen so dass daraus entsprechende Test-Ablöufe generiert werden können.

**Schnittstelle von Client- und Server-Entwicklung**

Der Client schickt an den Server REST-Requests (https Get bzw. Post) deren Daten der Server verarbeitet und daraufhin eine Response-Seite ausliefert.

Für jeden Aufruf gibt es im Modell ein Signal das durch Anclicken eines Buttons ausgelöst wird. Welcher Button welches Signal auslöst ist in den Win32-Wireframe--Diagrammen und in den Tags *OnClick* der Buttons dokumentiert.

*Beispiel:*

Im Dialog *Startseite* gibt es in der übergeordneten Gruppe *Register* ein Eingabefeld *Code* und den Button *Registrieren* der das Signal *StartRegistration* sendet. Entsprechend hat dieses Signal auch das Attribut *Code* das im Modell als *dialog.Startseite.Code* adressiert wird. Dementsprechend soll der Client bei Clicken des entsprechenden Buttons ein Post-Request zu *<BaseURL>/StartRegistration?Code=Wert des Eingabefeldes* abschicken, das dann vom Router auf dem Server an die Methode *Anmelder.startRegistration(Code)* des Controllers weitergegeben wird.

# Prozesse

## Initialisierung

Die Diagramme in diesem Abschnitt visualisieren die Abläufe der Interaktionen mit dem System. Sie enthalten Verweise auf die verwendeten Bildschirmformulare, die im nächsten Abschnitt beschrieben werden.



## Systemzugang



## Registrierung

Der Registrierungsprozess erfasst die Patientendaten und die Zustimmung des Patienten





## Hauptprozess



## Schreibaufgabe bearbeiten/Patientenstatus



## Erinnerungen/Timer

Der Timer führt um Mitternacht die folgenden Aktionen durch.

1. Registrierte Benutzer für die die Entlassungszeit eingegeben ist und für die diese Zeit erreicht ist erhalten die erste Schreibaufgabe. Der Status des Benutzers und der Aufgabe werden aktualisiert.
2. Die vom Therapeuten gestellte Aufgabe wird für den Patienten freigeschaltet wenn dessen Schreibtag erreicht ist. Der Status der Aufgabe und des Patienten wird aktualisiert und der Patient wird per Mail benachrichtigt.
3. Patienten, deren aktuelle Aufgabe 2 Tage nach Aufgabenstellung nicht abgeschickt ist, erhalten einmalig eine Erinnerungsnachricht. Der Status des Patienten und der Aufgabe werden aktualisiert.
4. Aufgaben die 5 Tage nach Aufgabenstellung nicht abgeschickt wurden werden als überfällig markiert. Der Status der Aufgabe und des Patienten wird aktualisiert.
5. Patienten werden 12 Wochen nach Erteilung der ersten Schreibaufgabe aus der Intervention herausgenommen. Ihr Status wird angepasst und für den verwendeten Wochentag wird ein Slot freigegeben.

**Diagramm Timer**



1. Timer

## Profil bearbeiten





## Tagebuch führen





## Ersten Tagebucheintrag schreiben

Die erste Schreibaufgabe weicht von den restlichen 11 Schreibaufgaben ab.

Textfeld 1 mit Aufforderung eine Situation bzgl. der Rückkehr an den Arbeitsplatz zu beschreiben.

Dann 3 kleine Textfelder mit der Aufforderung, die Situation aus Textfeld 1 nochmals zu betrachten und:

* in das erste der 3 kleinen Textfelder den Wunsch ans Gegenüber,
* in das zweite die Reaktion der anderen und
* ins dritte die Reaktion des Selbst zu schreiben.

Für zwei weitere Situationen müssen dann noch weitere Eingabemöglichkeiten gegeben sein.



## Weitere Tagebucheinträge schreiben

Textfeld ist erst klein (3-4 Zeilen) und wächst mit der Eingabe des Patienten (z.B. jQuery autogrow). So reduziert man die "Angst vor dem weißen Blatt Papier" (Schreibblockade) beim Patienten und vermeidet doppelte Scrollbalken, womit die Usability stark verbessert wird.

Die drei Kernfragen (Was waren Ihre Wünsche und Erwartungen in der Situation? Wie haben die anderen in der Situation reagiert? Wie haben Sie auf die anderen reagiert?) sind für den Teilnehmer während des Tagebuchschreibens sichtbar.



## Tagebücher kommentieren

Im Zentrum der Intervention soll die strukturierte Schreibaufgabe mit therapeutischer Kommentierung stehen.

Die Patientenschilderungen (Tagebücher) werden von einem für den Teilnehmer anonymen Online-Therapeuten schriftlich nach dem Modell der Supportiv-Expressiven Therapie (SET; Luborsky, 1984) kommentiert. Online-Therapeuten sind auf diese Form der Online-Kommentierung geschulte Mitarbeiter (Dipl.-/ M.Sc. Psychologen) des Studienzentrums.

Bei Einhaltung des jeweiligen Schreibtermins wird eine therapeutische Kommentierung innerhalb von 48 Stunden garantiert.

Der wöchentliche Zeitaufwand eines aktiven Studienteilnehmers kann auf etwa eine Stunde geschätzt werden, der Zeitaufwand für den geschulten Online-Therapeuten auf ca. 20 Minuten pro Rehabilitand.



# View (Formularentwürfe/Wireframes)

Die Diagramme in diesem Abschnitt zeigen die Entwürfe der Bildschirmseiten des Systems. Dabei ist vermerkt, welche Requests bei Auslösen einer Interaktion durch den Benutzer an den Server geschickt werden und welche Controller-Methoden diese Requests nach dem Routing bearbeiten.

Die Seiten sind als Blade-templates im Verzeichnis resources/views implementiert.

## Admin-Formular



**Codes:** Zeigt die Liste aller Codes mit ihrem Status (registriert/unregistriert)

**Patienten:** Zeigt die Liste aller Patienten mit ihrem Status, ermöglicht das Löschen einzelner Patienten und ihrer Daten

Reports: Erlaubt die Auswahl eines Reports aus einer Liste und den Export des Reports. Nicht implementiert!

**Therapeuten**: Zeigt die Liste aller Therapeuten. Ermöglicht das Löschen einzelner Therapeuten und das Anlegen neuer Therapeuten

## Anmeldeformulare





## Tagebucheintrag

Das Formular *entry* für den Tagebucheintrag wird sowohl von Patienten als auch von Therapeuten verwendet, wobei jeweils unterschiedliche Felder angezeigt werden.

* Das **Notiz-Feld** ($PatientInfo['notes']) ermöglicht dem Therapeuten die Eingabe zusätzlicher Informationen. Es wird für Patienten niemals angezeigt. Für Therapeuten ist es immer editierbar.
* Die Fragestellung (**Problem**, $EntryInfo['problem']) wird immer angezeigt.Für Patienten ist sie nicht editierbar. Für Therapeuten ist die Fragestellung nur editierbar wenn die Aufgabe die aktuelle Aufgabe ist ($EntryInfo['week'] == $PatientInfo['patientWeek']) und sie vom System noch nicht abgeschickt wurde ($EntryInfo['status'] < 'E020').
* **Antwort**: Answer of patient on problem. This can be for week == 1: array of situations for week > 1: string. Für Patienten ist der zuletzt gespeicherte, automatisch gespeicherte oder abgeschickte Inhalt (content, $EntryInfo['answer']) immer sichtbar aber nur editierbar wenn er nicht abgeschickt oder überfällig ist ($EntryInfo['status'] < 'E040'). Für Therapeuten ist der Inhalt nur sichtbar wenn er abgeschickt wurde ($EntryInfo['status'] >= 'E040'). Er ist für Therapeuten niemals editierbar.
* Für den Patienten werden die **Befindensfragen** (survey, $EntryInfo['survey']) nur angezeigt, wenn der Eintrag weder überfällig noch abgeschickt ist ($EntryInfo['status'] < 'E040'). Sie sind dann editierbar, d.h. sie können beantwortet werden. Für Therapeuten werden die Befindensfragen (survey) mit Antworten immer angezeigt. Sie sind nicht editierbar.
* **Kommentar** des Therapeuten: {{ $EntryInfo['comment'] }} Für Patienten ist der Kommentar (comment, EntryInfo->comment()) nur sichtbar wenn er vom Therapeuten abgeschickt wurde ($EntyInfo['status']>= 'E050'). Er ist für Patienten niemals editierbar. Für Therapeuten ist der Kommentar immer sichtbar, ggf. in einer zwischengespeicherten Version. Für Therapeuten ist der Kommentar nur editierbar wenn der Eintrag vom Patienten abgeschickt aber der Kommentar vom Therapeuten noch nicht abgeschickt ist. ($EntryInfo['status'] == 'E040')
* **Bewertung des Therapeutenkommentars** {{ $EntryInfo['comment\_reply'] }} Der Patient kann über die Kommentar-Rückmeldung (comment\_reply, $EntryInfo['comment\_reply']) einmalig dasNiveau seiner Zufriedenheit mit dem Kommentar eingeben. Das Feld wird für den Patienten immer dann angezeigt, wenn die Aufgabe kommentiert wurde. In diesem Fall kann der Patient die Rückmeldung eingeben und abschicken.($EntryInfo['status'] == 'E050') Ansonsten wird das Feld nicht angezeigt. Wird das Feld angezeigt, so soll der Patient nachdrücklich aufgefordert werden es auszufüllen. Der Therapeut sieht die Kommentar-Rückmeldung niemals.
* Die **Fragen-Gruppe** wird nur für Therapeuten und nur für die aktuelle oder künftige Aufgabe (Entry->base\_info()->week() >= PatientInfo->patientWeek) angezeigt wenn die Aufgabe noch nicht gestellt ist (EntryInfo->base\_info()->status() < 'E020') . Sie zeigt dieTitel der verfügbaren Fragen (Problems) an.. Ansonsten wird die Gruppe nicht angezeigt.
* Mit **choose\_template** kann der Therapeut eine vorgefertigte Frage wählen. Der Text des Frage wird in das Fregefeld (problem) übernommen, wo er editiert werden kann.
* Mit **new\_assignment** kann der Therapeut aus der aktuell eingegebenen Frage - nach Eingabe eines Titels - eine neue Standardfrage erstellen.
* **Zwischenspeichern** speichert die editierte Frage, den editierten Inhalt oder Kommentar auf dem Server. Bei einem erneuten Aufruf der Seite werden content für den Patienten) bzw. problem oder comment (für den Therapeuten) damit vorbelegt. Danach kehrt der Benutzer zur selben Seite zurück.
* **Abschicken** schickt alle editierbaren Formularfelder an den Server. Der Status des Benutzers und des Eintrags werden auf dem Server angepasst. Der Benutzer wird zu seiner Homepage weitergeleitet.
* "**Älter**" führt zum vorherigen Eintrag, "**Neuer**" zum folgenden. Beide Buttons werden nur angezeigt, wenn es einen vorhergehenden bzw. folgenden Eintrag gibt (EntryInfo->base\_info()->week() > 1 bzw. EntryInfo->base\_info()->week()<12).
* "**Zur Übersicht**" leitet zur Homepage des Patienten (diary) bzw. des Therapeuten (patient\_list).
* Ist die angezeigte Aufgabe nicht die aktuelle Aufgabe und befindet sich der Patient noch inerhalb der Studie, so gibt es den Button "**Aktuelle Aufgabe**". Er führt *z*ur aktuellen Schreibaufgabe.



## Tagebuch

Seite mit der Liste aller Einträge des Tagebuchs eines Patienten. Dies ist die Homepage für Patienten. Therapeuten können über die Patientenliste darauf und damit auf das Profil und auf die Tagebucheinträge zugreifen.

Keines der Elemente dieser Seite lässt sich bearbeiten.

* Der Fortschritt zeigt an, die wievielte der 12 Wochen der Intervention die aktuelle Woche ist. Er wird immer angezeigt.
* name ist der Benutzername des Patienten
* Profil führt zum Profil des Patienten
* Abmelden ermöglicht ein Beenden der laufenden Sitzung
* Die Gruppe current\_entry enthält Angaben zur aktuellen Schreibaufgabe, falls eine existiert, insbesondere Wochennr. (week\_nr) und Status (entry\_status). problem ist die Aufgabenstellung
* Die Gruppe closed\_entry (mehrfach vorhanden) enthältdie entsprechenden Angaben zu abgeschlossenen (d.h. kommentierten oder überfälligen) Aufgaben.
* Bearbeiten und Ansehen führt zur Seite entry für den entsprechenden Beitrag
* Wochenübersicht führt zu einer Seite mit allen Aufgaben, Antworten und Kommentaren, die ggf. im Browser als pdf gespeichert oder ausgedruckt werden kann.



## Wochenübersicht

Die Wochenübersicht enthält die grafische Darstellung der Entwicklung der Selbsteinschätzung sowie alle bisherigen Schreibaufgaben, Kommentare und Bewertungen. Das Formular wird sowohl von Patienten als auch von Therapeuten verwendet.

Ein Diagramm ist für dieses Formular nicht verfügbar.

## Patientenprofil

Das Patientenprofil wird für Patienten und Therapeuten mit unterschiedlichen Templates realisiert. Es besteht aus mehreren Formularen.



## Patientenliste

Liste aller Patienten.

Dies ist die erste Seite, die Therapeuten nach der Anmeldung sehen.

In der Gruppe Slots kann die Zahl der freien Slots pro Tag überprüft und ggf. erhöht werden.

Es gibt einen festen Tag für die Schreibaufgabe. Diesen Tag wählt der Patient. Der Patient wird dann an die Aufgabe per Mail erinnert.

Der Schreibtag ist nachträglich ein mal wechselbar.

Für jeden Tag gibt es es eine "Obergrenze" an Teilnehmern geben, die sich zuordnen können und ansonsten einen Hinweis à la "Dieser Schreibtag ist bereits voll belegt, bitte wählen Sie einen anderen Tag aus" bekommen, um zu vermeiden, dass alle sich den gleichen Tag auswählen und die Therapeuten die Tagebucheinträge nicht innerhalb der zugsagten 24 h bearbeiten können.

Wir gehen von einer Mindest-Bearbeitungszeit von 20 Minuten pro Tagebucheintrag aus. Wenn man das auf die Arbeitszeiten hochrechnet, liegt das absolute Maximum von zu bearbeitenden Tagebucheinträgen pro Tag bei 30. Das sollte also die Höchstzahl von möglichen Zuweisungen pro Tag sein.

* In der Gruppe Filter kann die Liste nach den Werten in einer Spalte gefiltert werden
* Die Bedeutung der Spalten ist in den Notes zur Klasse PatientListEntry definiert.
* Durch Click auf einen Spaltenkopf kann nach den Werten in dieser Spalte sortiert werden
* Durch Click auf den Namen eines Patienten wird die Seite mit dem Tagebuch und den Daten dieses Patienten zur Bearbeitung aufgerufen.
* Außerdem können Patienten für Massenaktionen (Mail) ausgewählt werden.
* Man kann sich von der Seite aus ausloggen



## Maileditor

Mit dem Maileditor kann eine Mail an mehrere Patienten verschickt werden, die in der Patientenliste ausgewählt wurden.



# Controller

Die Klassen des Controllers implementieren die Funktionalität des Systems. Der Code dieser Klassen befindet sich im Verzeichnissen *app/http/Controllers*. Hilfsklassen befinden sich im Verzeichnis *app*. Die Controller-Klassen greifen auf die Daten des Datenmodells zu, das im nächsten Abschnitt beschrieben wird. Der angegebene Pseudocode dient nur der Orientierung.



## AdminController

| OPERATIONS |
| --- |
| admin\_codes () : Public  Zeigt die Liste aller Codes mit ihrem Status (registriert/unregistriert) |
| admin\_users () : Public  Zeigt die Liste aller Benutzer mit ihrer Rolle |

## AuxController

| OPERATIONS |
| --- |
| home () : Public  Rücksprung zur Homepage des Benutzers. Dies ist   * für angemeldete Patienten die Seite mit der Tagebuchübersicht * für angemeldete Therapeuten die Seite patient\_list * für alle anderen die Basis-URL /, also die jeweilige Startseite   Dazu muss die Rolle des Benutzers im Objekt user in Verbindung mit dem Session key auf dem Server gespeichert werden.  Behavior: if (session\_info.role == 'therapist') {  patient\_list = new Patient\_list;  patient\_list.show(session\_info.page\_definition);  } else if (session\_info.role == 'patient')  {  diary.show();  } else {Anmelder.enter\_system(Cookie);  } |
| logout () : Public  Beendet die Session und letet zur rollenspezifischen Login-Seite mit einer Bestätigung der erfolgreichen Abmeldung. StayLoggedIn wird auf false gesetzt. |

## ContactController

| OPERATIONS |
| --- |
| contact\_team () : Public  Zeige das Kontaktformular  Behavior: return view('system.contact\_form'); |
| mail\_editor (list\_of\_patients : var ) : Public  Zeigt ein Formular zum verfassen von Mails an eine reihe von Patienten an |
| message\_to\_patients (list\_of\_names : var , mail\_subject : var , mail\_body : var ) : Public  Sendet eine Mail an eine Liste von Patienten |
| send\_message (eMail : , subject : , message : ) : Public  Die Nachricht mit angegebenem subject, Absender und message wird an das team geschickt und  Es wird mit alert bestätigt, dass eine Nachricht an das Team geschickt wurde.  Behavior: Send Message to team;  Alert("Ihre Nachricht wurde an das Projektteam übermittelt");  Redirect::to('/Home); |

## Days

| ATTRIBUTES |
| --- |
| days\_map : var Private  [name] = number |

| OPERATIONS |
| --- |
| \_\_construct () : var Public |
| day\_available () : var Public  Es wird true zurückgegeben wenn es wenigstens einen Tag gibt, für den eine Registrierung möglich ist, sonst false  Behavior: if (get\_available\_days() == {})  return false;  } else {  return true;  } |
| decrease\_day (day : var ) : var Public  Die Anzahl der Slots für den Tag wird um eins vermindert. Wenn kein Slot übrig ist wird der Admin benachrichtigt. |
| get\_available\_days () : var Public  Es wird die Liste aller der Tage T zurückgegeben für die es wenigstens einen freien Slot gibt. Die Liste hat die Form ["Sonntag", ... , "Donnerstag"].  Behavior: var day\_list=get\_available\_days();  var available\_days={};  for each day in keys(day\_list){  if (day\_list.day > 0) {  push(day,available\_days);  }}  return day\_list; |
| get\_days () : var Public  Es wird eine Liste aus der Datenbank geholt die angibt, für welchen Tag wieviele Slots verfügbar sind. Der Rückgabewert hat die Form einer Liste wie ["Sonntag" => 3, ... , "Donnerstag" => 5] . In der Liste sind alle Tage von Sonntag bis Donnerstag genau einmal vertreten. |
| get\_week\_day (day : var ) : var Private  Reads the weekday's entry and returns the eloquent model.  @param $day the weekday - either a number between 0 and 6 or the name  @return the appropriate eloquent model |
| send\_mail (day : WeekDay ) : var Private  Den Admin via E-Mail darüber informieren, dass alle Plätze belegt sind.  @param WeekDay $day |
| set\_days (day\_list : var ) : var Public  Für jeden Tag wird in der Datenbank die Anzahl der verfügbaren Tage gespeichert. Das Argument wird in der Form einer Liste der Form ["Sonntag"=>3, ... ,"Donnerstag"=>5] übergeben. In der Liste sind alle Tage von Sonntag bis Donnerstag genau einmal vertreten. |

## DiaryController

| OPERATIONS |
| --- |
| add\_situation () : Public  Füge eine neue Situation zur ersten Schreibaufgabe hinzu |
| commented\_diary () : int Public  Zeige alle bisherigen Beiträge sowie die dazu erfolgten Kommentare auf einer Seite an, die vom Browser gespeichert oder ausgedruckt werden kann. |
| entry (WeekNr : int ) : Public  Es wird der zur als Argument übergebenen Wochennummer gehörende Tagebucheintrag für den Patienten mit Name name ausgegeben. Soweit ein Kommentar vorhanden ist wird er mit ausgegeben.  Für Patienten und Therapeuten werden unterschiedliche Ansichten ausgegeben (s. Notes zu Wireframe )  Für Patienten:  Je nach Art der Aufgabe und Status des Patienten kann der Text editiert werden oder nicht:  P030, P040, P080, P090: Editierbar  Sonst nicht editierbar |
| get\_response (entry\_id : int ) : Public  Die Seite mit dem Eintrag zur übergebenen Id wird angezeigt.  Es wird überprüft, ob die Id zum Patienten der aktuellen Session gehört. Ist das nicht der Fall so wird der Cookie gelöscht und es wird aud die Startseite weitergeleitet.  Je nach Status des Patienten wird die anzuzeigende Seite ansonsten gestaltet:  Ist nicht die aktuelle Aufgabe ausgewählt, so weirden die gewählte Aufgabe und der Kommentar nicht editierbar angezeigt.  Ist die aktuelle Aufgabe ausgewählt, so wird die aktuelle Aufgabe je nach Status des Patienten angezeigt. dabei werden unterschiedliche Seiten ausgeliefert, je nachdem ob es sich um die erste Aufgabe oder eine Folgeaufgabe handelt.  Die folgenden Fälle sind relevant (s. Patient\_status):   * Erste Aufgabe erhalten: Aufgabe editierbar * Erste Aufgabe bearbeitet: Aufgabe editierbar mit zwischengespeichertem Inhalt * Erste Aufgabe abgeschickt: Aufgabe nicht editierbar und Antwort * Erste Aufgabe kommentiert: Aufgabe und Antwort nicht editierbar mit Kommentar * Erste Aufgabe versäumt: Aufgabe nicht editierbar und Hinweis auf Versäumnis * Aktuelle Folgeaufgabe erhalten: Aufgabe editierbar * Aktuelle Folgeaufgabe bearbeitet: Aufgabe editierbar mit zwischengespeichertem Inhalt * Aktuelle Folgeaufgabe abgeschickt: Aufgabe und Antwort nicht editierbar * Aktuelle Folgeaufgabe kommentiert: Aufgabe und Antwort nicht editierbar mit Kommentar * Aktuelle Folgeaufgabe versäumt: Aufgabe nicht editierbar und Hinweis auf Versäumnis   Behavior: if (not actual assignment) {  return view(diary.entry\_noneditable)-> where('Content'="Complete content", Comment="Comment");  Result: Not Actual  } else if (first assignment) {  Result: First  } else {  Result: Successive  } |
| save\_entry (patient : var , week : int , problem : var , content : var , survey : var , comment : var , comment\_reply : var ) : Public  Speichern der Änderungen zur Frage und/oder Antwort in Abhängigkeit vom gewählten Button im entry view (Wert von entryButton) |
| select\_assignment (assignment\_id : int ) : int Public  Wähle die Aufgabe mit dem angegebenen Titel aus und trage den Aufgaben-text in entry.problem ein |
| show ($name : var ) : Public  Zeigt das Tagebuch des Patienten mit dem Benutzernamen name  Behavior: // Setting default parameter  if (! $name) {  $name = Auth::user()-> name;  }  /\*\*  \* If the user is a patient, he can only see his own diary  \*/  if (get\_class($request->user()->userable) == 'App\Patient' && Auth::user()->name != $name) {  return Redirect::to('/');  }  // return $name;  return view('patient.diary')->with('name',$name); |

## DiaryEntry

| ATTRIBUTES |
| --- |
| Id : int Public  An Id which is unique for all entries for all patients |
| Problem : Public |

## GateController

Diese Klasse behandelt alle Aufrufe des Servers in Zusammenhang mit dem Registrierungs- und Anmeldeprozess. Neben den angegebenen Operationen wird für jedes Signal eine Methode benötigt die den entsprechenden https-Aufruf mit den Signalparametern verarbeitet.

| OPERATIONS |
| --- |
| \_\_construct () : var Public |
| \_\_destruct () : var Public  [ Is static False. Is abstract False. Is return array False. Is query False. Is synchronized False. ] |
| check\_login\_password (NameOrEmail : String , Password : String , StayLoggedIn : boolean ) : var Public  wenn Username oder Password falsch sind wird Zugriff verweigert. Ansonsten wird Zugriff erlaubt und Code und StayLoggedin weirden im Cookie gespeichert.  Behavior: if (incorrect name or password) {  Result: AccessAllowed=false;  return View::make(system.info\_message) -> where ('Text',"Falscher Benutzername oder Passwort, bitte noch einmal versuchen");  } else {  Result: AccesAllowed=true;;  Store Code and Cookie.code  Store StayLoggedIn in Cookie.stay\_loggedin;  Create SessionInfo;  AuxController@home();  } |
| code\_status (Code : String ) : var Private  Behavior: if (Code == "BBB") {  return "registered;  } else if (Code == "AAA") {  return "unregistered";  } else {  return "incorrect";  } |
| enter\_system (cookie : Cookie ) : var Public  Beim ersten Aufruf wird geprüft   1. Ob der Cookie gesetzt ist 2. Wenn ja ob der Parameter StayLoggedIn auf true steht 3. Es wird geprüft, ob der Code im Cookie registriert ist. 4. Schlägt einer dieser Tests fehl wird verfahren als ob der Cookie nicht gesetzt wäre. 5. Ist StayLoggindIn==true so wird zum Tagebuch schreiben weitergeleitet. 6. Gibt es keine freien Tage so wird so wird die Seite Login\_only ausgeliefert 7. Ansonsten wird die Seite Startseite ausgeliefert |
| from\_welcome () : var Public  Weiterleitung zur Seite mit den Verpflichtungen  Behavior: return view(gate.accept); |
| get\_reset\_code (ResetCode : String ) : var Public  Behavior: if (ResetCode == "BBB-Reset") {  Trace("Zur Profilseite");  sim.ResetCodeCorrect=true;  } else {  dialog.ResetCodeIncorrect.Show=true;;  } |
| mail\_for\_password (Mail : String ) : var Public  Es wird geprüft, ob die Mail-Adresse registriert ist. Wenn ja wird ein Reset- Code dafür angelegt, eine Mail wird verschickt. Um die vorhandenen Mailadressen zu schützen wird der Benutzer In jedem Fall darüber informiert, dass eine Mail verschickt wurde, auch wenn dies nicht der Fall ist, weil die Mail-Adresse nicht registriert war.  Behavior: if (Mailadresse registriert)  send mail to Mail  Alert::info('Es wurde eine Mail an '.Mail.' verschickt.');  return redirect('/'); |
| registration\_complete () : var Public  Behavior: dialog.Startseite.show=true; |
| req\_patient\_data () : var Public  Auf dem Server werden die verfügbaren Tage berechnet: Days.get\_available\_days() Wenn keine Tage verfügbar sind wird zur Startseite zurückgesprungen die dann die Information anzeigt, dass eine Registrierung nicht möglich ist. Mit diesen Informationen wird die Seite zur Erfassung der Patientendaten aufgebaut und ausgeliefert.  Behavior: if (! Days.days\_available()) {  Generic.home();  return;  }  Setze Auswahlliste Patientendaten.Wochentag unter Verwendung von Days.get\_available\_days()  Zeige Seite PatientenDaten |
| reset\_password () : var Public  Behavior: dialog.PasswortVergessen.Show=true; |
| save\_patient\_data (Code : var , Name : var , Password : var , eMail : var , Day : var ) : var Public  Es werden im Profil des Patienten gespeichert:   1. Cookie.Code 2. Name 3. Passwort1 4. Email1 5. gewählter Tag 6. Zeit der Registrierung   Es wird vermerkt, dass der Code registriert ist Die Anzahl der Slots für den gewählten Tag wird um 1 vermindert: Days.decrease\_day(gewählter Tag)  Behavior: if (Name or eMail already in use) {  Result: Registered=false;  return View::make(system.info-message) -> with('text',"Ihre Registrierung ist lleider fehlgeschlagen, Bitte wählen Sie einen anderen Benutzernamen");  } else {  Result: Registered=true;  Patient.code=Code;  Patient.user\_name=Name;  Patient.assword=Password;  Patient.eMail=eMail;  Patient.day=Day;  save Patient;  confirmation\_message ''registration\_success';  redirect /Home  } |
| start\_registration (Code : String ) : var Public  Start des Registrierungsprozesses. System prüft, ob der Code in der Datenbank vorhanden und noch nicht belegt ist. Die Begrüßungsseite wird im Erfolgsfall angezeigt. Ansonsten wird der Benutzer informiert  Behavior: if (Cookie.code already registered) {  return View::make(system.info\_message) -> where ('Text',"Dieser Code wurde bereits registriert, Sie können sich anmelden");  Result: CodeStatus="registered";  } else if (Code not yet registered) {  return view(gate.welcome);  Result: CodeStatus="unregistered";  } else {  return View::make(system.info\_message) -> where ('Text',"Der einegegebene Code ist nicht korrekt. Hilfe zur Code-Eingabe:...");  result: CodeStatus="incorrect";  }  return sim.CodeStatus; |

## PatientController

Die Klasse zeigt das Profil des Patienten an und erlaubt Veränderungen daran.

| OPERATIONS |
| --- |
| cancel\_intervention (name : var ) : Public  Bricht die Intervention für den Patienten mit dem angegebenen Benutzernamen ab,  Behavior: $patient=Patient(name);  $patient->patientStatus="P130";  Save $patient;  Alert('Zusammenarbeit mit Patient '.name.' beendet');  return Redirec::to('/diary/'.name); |
| profile (name : var ) : var Public  Zeigt die Profilseite des Patienten mit dem angegebenen Benutzernamen an. Ist name nicht angegeben, so muss die Rolle des benutzers 'patient' sein und es wird die Profilseite des aktuellen Patienten angezeigt.  Behavior: if (! $name) {  $name=Session::get(user->name);  }  $patient=Patient(name);  eturn view('patient.patient\_profile')->where('Patient'=>$patient, 'UserName' => name); |
| save\_profile (name : var , date\_from\_clinics : var , new\_password : var , notes : var , personal\_information : var , therapist : var , old\_password : var ) : Public  Das Profil wird mit den geänderten Daten aktualisiert. Bei Passwortänderung wird vorher geprüft ob das alte Passwort korrekt ist. Anschließend wird zur Homepage des Benutzers weitergeleitet.  Behavior: $patient=Patient(name);  if ($request->'oldPassword' != Password of name) {  alert('Falsches Passwort.');  return view('patient.patient\_profile')->where('UserName'=>$name, 'Patient'=>$patient);  }  foreach ($par as args) {  this.change\_in\_profile($patient,$par, $request[$par]);  }  save $patient to database;  alert("Profil für ".$name." aktualisiert.");  Redirect::to('/Home'); |

## PatientListController

Diese Klasse implementiert alle Aktionen, die der Therapeut auf der Patientenliste vornehmen kann

* Freie Slots setzen,
* Liste sortieren und
* Filtern

| OPERATIONS |
| --- |
| change\_in\_profile (patient : Patient , field : , value : ) : int Public  Ändern eines Werts im Patientenprofil  Behavior: switch ($field) {  case ''assignmentDay':  if ($field != $patient->'assignmentDay') {  $patient->'assignmentDay' = value;  $patient->assignmentDayChangesLeft--;  break;  }  case 'dateFromClinics'  $patient->'dateFromClinics'=$value;  $patientStatus update  } |
| set\_slots (days : Days ) : Public  Days wird im System modifiziert und die Seite mit der Patientenliste (der der Slots-Teil davon) wird neu aufgebaut  Behavior: page\_definition=this.old\_page\_definition();  $days=new Days;  $days->set\_days(Days);  this.show(page\_definition; |
| show (page\_definition : Page\_definition ) : Public  'Liefere Seite patient\_list'(   * 'Slots'(Days) * 'Filter'(page\_definition) * patients(   this.patients(page\_definition)   * page\_definition   )  Seite setzt Cookie.last\_url="patient\_list/show?'page\_definition'=page\_definition"  Behavior: Zeige Seite patient\_list mit  Slots von Days,  Filter von page\_definition.filter,  Patientenliste von this.patients(page\_definition),  Session::put('PageDefinition',page\_definition);  return view('patient\_list') -> where Slots, Filter,PatientList |

## Timer

Der Timer führt um Mitternacht die folgenden Aktionen durch.

1. Registrierte Benutzer für die die Entlassungszeit eingegeben ist und für die diese Zeit erreicht ist erhalten die erste Schreibaufgabe. Der Status des Benutzers und der Aufgabe werden aktualisiert.
2. Die vom Therapeuten gestellte Aufgabe wird für den Patienten freigeschaltet wenn dessen Schreibtag erreicht ist. Der Status der Aufgabe und des Patienten wird aktualisiert und der Patient wird per Mail benachrichtigt.
3. Patienten, deren aktuelle Aufgabe 3 Tage nach Aufgabenstellung nicht abgeschickt ist, erhalten einmalig eine Erinnerungsnachricht. Der Status des Patienten und der Aufgabe werden aktualisiert.
4. Aufgaben die 7 Tage nach Aufgabenstellung nicht abgeschickt wurden werden als überfällig markiert. Der Status der Aufgabe und des Patienten wird aktualisiert.
5. Patienten werden 12 Wochen nach Erteilung der ersten Schreibaufgabe aus der Intervention herausgenommen. Ihr Status wird angepasst und für den verwendeten Wochentag wird ein Slot freigegeben.

| ELEMENTS OWNED BY Timer |
| --- |
| Verschicke Erinnerung dass Aufgabe bald fällig : Activity |
| Verschicke Information über abgelaufene Frist : Activity |
| Verschicke Information über erste Aufgabe : Activity |
| Verschicke Information über neue Aufgabe : Activity |
| 1. Woche? : DecisionNode |
| Aktuelle Aufgabe ist definiert und der Patient wurde darüber noch nicht informiert : DecisionNode |
| Bearbeitungsfrist abgelaufen und Patient ist noch nicht darüber informiert : DecisionNode |
| Mahnfrist für aktuelle Aufgabe ist erreicht und Patient ist noch nicht darüber informiert : DecisionNode |
| End : ActivityFinal |
| Start um 0:00 : ActivityInitial |

| OPERATIONS |
| --- |
| do\_timer () : Public  Does periodically all specified actions . Is triggered by timer event. |

### Timer diagram



# Model (Daten)

Dieser Abschnitt beschreibt wichtige Datenstrukturen, die in der Datenbank gespeichert werden.

## Constants / Basic Classes diagram



## Fragen nach dem Befinden



Es werden nur die Antworten benötigt, die Fragen stehen im Template (in der gerenderten View).

## Main Model diagram

Jedes Modell (wie Patient) hat eine Methode "to\_info()" die eine Beschreibung des Modells liefern kann.

Diese Beschreibung ist ein Array, dessen Werte an eine View übergeben werden können - d.h. die Werte liegen in einem aufbereiteten Format vor (jedes Datum ist z.B. in einen passenden String überführt worden).

Beschreibungen können zudem kombiniert werden, indem ein weiteres Array angegeben wird. Dieses wird dann mit den Informationen kombiniert (Achtung: das Ziel-Array wird nicht modifiziert, der Rückgabewert enthält das zusammengestellte Array).

Eine detaillierte Beschreibung gibt es unter

https://gitlab.uni-koblenz.de/iwm/gsa-online-plus/wikis/info-method-description



## Answer

| ATTRIBUTES |
| --- |
| answer : int Public |
| patient : Patient Public |

## AnswersPHQ4

| ATTRIBUTES |
| --- |
| AN\_EINZELNEN\_TAGEN : int Public Const = 1 |
| AN\_MEHR\_ALS\_DER\_HAELFTE\_DER\_TAGE : int Public Const = 2 |
| BEINAHE\_JEDEN\_TAG : int Public Const = 3 |
| UEBERHAUPT\_NICHT : int Public Const = 0 |

## AnswersReply

| ATTRIBUTES |
| --- |
| ETWAS : int Public Const = 2  [ Is static True. Containment is Not Specified. ] |
| GAR\_NICHT : int Public Const = 0  [ Is static True. Containment is Not Specified. ] |
| KAUM : int Public Const = 1  [ Is static True. Containment is Not Specified. ] |
| SEHR : int Public Const = 4  [ Is static True. Containment is Not Specified. ] |
| ZIEMLICH : int Public Const = 3  [ Is static True. Containment is Not Specified. ] |

## Assignment

| ATTRIBUTES |
| --- |
| dirty : boolean Public  Text lediglich zwischengespeichert = true  Text eingereicht / abgeschickt = false |
| week : int Public  Woche, in welcher die Aufgabe gestellt wurde |
| writing\_date : Date Public  Date on which the assignment starts.  Set when the patient receives the previous assignment. May also change if the patient decides to change the assignment date. |

| OPERATIONS |
| --- |
| status () : String Public  Status der Aufgabe.  Konkret: ein durch die Klasse AssignmentStatus vorgegebener String.  Dieser beinhaltet die Nummer (Exyz), welche mit dem Array AssignmentStatus::$STATUS\_INFO in eine Info übersetzt werden kann) |

## AssignmentStatus

| ATTRIBUTES |
| --- |
| ASSIGNMENT\_IS\_NOT\_DEFINED : Public = "E010"  [ Is static True. Containment is Not Specified. ] |
| ASSIGNMENT\_IS\_NOT\_REQUIRED : Public = "E100"  [ Is static True. Containment is Not Specified. ] |
| PATIENT\_EDITED\_ASSIGNMENT : Public = "E030"  [ Is static True. Containment is Not Specified. ] |
| PATIENT\_FINISHED\_ASSIGNMENT : Public = "E040"  [ Is static True. Containment is Not Specified. ] |
| PATIENT\_GOT\_ASSIGNMENT : Public = "E020"  [ Is static True. Containment is Not Specified. ] |
| PATIENT\_RATED\_COMMENT : Public = "E060"  [ Is static True. Containment is Not Specified. ] |
| STATUS\_INFO : array Public = array(  "E010"=> "E010: Aufgabe nicht definiert",  "E015" => "E015: Aufgabe definiert",  "E020" => "E020: Aufgabe gestellt",  "E030" => "E030: Aufgabe bearbeitet",  "E040" => "E040: Aufgabe abgeschickt",  "E050" => "E050: Antwort kommentiert",  "E060" => "E060: Kommentar bewertet",  "E070" => "E070: Aufgabe überfällig",  "E100" => "E100: Aufgabe nicht erforderlich"  ) |
| SYSTEM\_REMINDED\_OF\_ASSIGNMENT : Public = "E070" |
| THERAPIST\_COMMENTED\_ASSIGNMENT : Public = "E050" |
| THERAPIST\_SAVED\_ASSIGNMENT : Public = "E015" |
| UNKNOWN : Public = "E000" |

| OPERATIONS |
| --- |
| to\_patient\_status (assignment\_status : String ) : String Public  liefert den jeweiligen Patient-Status: ein durch die Klasse PatientStatus vorgegebener String.  Dieser beinhaltet die Nummer (Pxyz), welche mit dem Array PatientStatus::$STATUS\_INFO in eine Info übersetzt werden kann)  Gibt "P000" zurück, falls kein Äquivalent gefunden wird. |

## AssignmentTemplate

| ATTRIBUTES |
| --- |
| text : String Public |
| title : String Public |

## Code

| ATTRIBUTES |
| --- |
| value : String Public  ein Code |

## Comment

| ATTRIBUTES |
| --- |
| date : Date Public  Datum an welchem der Therapeut den Kommentar verfasst hat |
| text : String Public  Kommentar des Therapeuten |

## CommentReply

| ATTRIBUTES |
| --- |
| helpful : int Public  Wie hilfreich waren die Rückmeldungen des Online-Therapeuten? 0 1 2 3 4 |
| satisfied : int Public  Wie zufrieden waren Sie mit der Rückmeldung des Online-Therapeuten? 0 1 2 3 4 |

## DayOfWeek

Schreibtag kann aber ebenso Sonntag sein.

week of day ist ein beliebiger Wochentag,evtl. "DayOfWeek" als Klassennamen verwenden

| ATTRIBUTES |
| --- |
| free\_time\_slots : int Public  Anzahl der freien Plätze (für diesen Tag) |
| name : String Public  Name des Tages (in Deutsch - z.B. Montag) |
| number : int Public  Nummer des Tages (0 = Sonntag ... 6 = Samstag) |

## Patient

| ATTRIBUTES |
| --- |
| assignment\_day : int Public  der ausgewählte Schreibtag |
| assignment\_day\_changes\_left : int Public  verbleibende Änderungen des Schreibtags |
| code : String Public  Registrierungscode. Wird von Therapeuten benötigt um Entlassungsdatum einzugeben |
| date\_from\_clinics : Date Public  Entlassungstag |
| intervention\_ended\_on : Date Public  Ende der Intervention (Datum) |
| last\_activity : Date Public  Datum der letzten Aktivität |
| last\_login : Date Public |
| notes\_of\_therapist : String Public  Notizen des Therapeuten |
| personal\_information : String Public  Persönliche Informationen die der Patient freiwillig bereitstellt |
| registration\_date : Date Public  wann der Patient sich registriert hat |

| OPERATIONS |
| --- |
| assignment\_for\_week (week : int ) : Assignment Public  Aufgabe, die in der angegebenen Woche aktuell war / ist |
| current\_assignment () : Assignment Public  die aktuelle Aufgabe |
| first\_assignment () : SituationSurvey Public  Antwort auf die erste Frage (eine Beschreibung von verschiedenen Situationen am Arbeitsplatz) |
| first\_assignment\_day () : Date Public  der erste Schreibtag (Tag der ersten Aufgabe) |
| intervention\_ended\_in\_week () : int Public  die Woche, in welcher die Intervention beendet wurde |
| next\_assignment () : Assignment Public  die nächste Aufgabe |
| ordered\_assignments () : Collection Public  sortierte Liste aller Aufgaben |
| overdue () : float Public  Anteil der überfälligen Einträge in Prozent (Anzahl überfällige + versäumte Aufgaben / Anzahl aller Aufgaben) |
| past\_assignments () : Collection Public  eine Liste der vorherigen Aufgaben |
| past\_assignments\_without\_comment () : Collection Public  vorige Aufgaben die noch nicht kommentiert wurden |
| patient\_week () : int Public  Nummer der Woche der Intervention (-1...12)  -1: Patient ist noch in der Klinik  0: Patient hat die Klinik verlassen, der erste Schreibtag liegt noch in der Zukunft  n: die n-the Woche |
| previous\_assignment\_day () : Date Public  der letzte Schreibtag (Tag der letzten Aufgabe) |
| status () : String Public  Status des Patienten. Für registrierte Patienten in der Interventionsphase ist das der Status der aktuellen Aufgabe.  Konkret: ein durch die Klasse PatientStatus vorgegebener String.  Dieser beinhaltet die Nummer (Pxyz), welche mit dem Array PatientStatus::$STATUS\_INFO in eine Info übersetzt werden kann) |
| status\_of\_next\_assignment () : String Public  Liefert den Status der Schreibaufgabe der kommenden Woche.  Konkret: ein durch die Klasse AssignmentStatus vorgegebener String.  Dieser beinhaltet die Nummer (Pxyz), welche mit dem Array AssignmentStatus ::$STATUS\_INFO in eine Info übersetzt werden kann) |
| week\_for\_date (date : Date ) : int Public  berechnet die zum Datum gehörende Woche (Beispiel: Tag nach dem ersten Schreibtag -> 1-te Woche) |

## PatientStatus

| ATTRIBUTES |
| --- |
| COLLABORATION\_ENDED : Public = "P130" |
| DATE\_OF\_DEPARTURE\_SET : Public = "P025" |
| INTERVENTION\_ENDED : Public = "P140" |
| PATIENT\_EDITED\_ASSIGNMENT : Public = "P040" |
| PATIENT\_FINISHED\_ASSIGNMENT : Public = "P050" |
| PATIENT\_GOT\_ASSIGNMENT : Public = "P030" |
| PATIENT\_RATED\_COMMENT : Public = "P065" |
| REGISTERED : Public = "P020" |
| STATUS\_INFO : Public = array(  "P000" => "P000: Unbekannt",  "P020" => "P020: Registriert",  "P025" => "P025: Entlassungsdatum erfasst",  "P030" => "P030: Aufgabe erhalten",  "P040" => "P040: Aufgabe bearbeitet",  "P045" => "P045: Aufgabe gemahnt",  "P050" => "P050: Aufgabe abgeschickt",  "P060" => "P060: Aufgabe kommentiert",  "P065" => "P065: Aufgabenkommentar bewertet",  "P130" => "P130: Mitarbeit beendet",  "P140" => "P140: Interventionszeit beendet"  ) |
| SYSTEM\_REMINDED\_OF\_ASSIGNMENT : Public = "P045" |
| THERAPIST\_COMMENTED\_ASSIGNMENT : Public = "P060" |
| UNKNOWN : Public = "P000" |
| UNREGISTERED : Public = "P010" |

## Question

| ATTRIBUTES |
| --- |
| id : String Public |
| question\_text : String Public |

## QuestionsIDs

| ATTRIBUTES |
| --- |
| PHQ4 : String Public |
| PHQ4\_DEPRESSED : String Public |
| PHQ4\_INTERESTED : String Public |
| PHQ4\_NERVOUS : String Public |
| PHQ4\_TROUBLED : String Public |
| REPLY : String Public |
| REPLY\_HELPFUL : String Public |
| REPLY\_SATISFIED : String Public |
| WAI : String Public |
| WAI\_INDEX : String Public |

## Situation

| ATTRIBUTES |
| --- |
| description : String Public  description of situation |
| expectation : String Public  Expectation of patient to situation |
| my\_reaction : String Public  reaction of patient in situation |
| their\_reaction : String Public  reaction of others in situation |

## Survey

| ATTRIBUTES |
| --- |
| health : int Public  Antwort auf Frage zur Einschätzung des Gesundheitszustands, Werte -1,...,10, default für nichtbeantwortete Fragen ist -1 |
| wai : int Public  Antwort auf wai-Frage, Werte -1,...,10, default für nichtbeantwortete Fragen ist -1 |

## Survey

| ATTRIBUTES |
| --- |
| id : String Public |
| title : String Public |

| OPERATIONS |
| --- |
| find\_question (questionID : String ) : Question Public |

## Task

| ATTRIBUTES |
| --- |
| answer : String Public  Answer of patient on problem. |
| problem : String Public  Problem to be reflected on (Aufgabenstellung) |

## TaskTemplate

| ATTRIBUTES |
| --- |
| name : String Public  Title der Vorlage |
| problem : String Public  Text der Vorlage (Problemstellung - siehe Task.problem) |

## User

| ATTRIBUTES |
| --- |
| email : String Public  E-Mail-Adresse |
| is\_random : boolean Public  ob der Patient durch den Seeder generiert wurde  true = durch den Seeder erzeugter Testpatient  false = Nutzer, der sich über das System registriert hat |
| name : String Public  Username |
| password : String Public  Passwort |