Entwurfsbeschreibung

1. Allgemeines

Ziel unseres Projekts ist es den OLAT-Kalender um einige nützliche Funktionen zu erweitern.
Die Kalendererweiterung soll den Nutzern des OLAT mehr Möglichkeiten geben, ihre Termine zu verwalten sowie mehr Komfort bei der Einstellung der Termine bieten. Die Erweiterung wird in die Onlineanwendung OLAT eingebettet. OLAT (inklusive unserer Erweiterung) soll auf Servern der Universität installiert werden. Ein Zugriff auf OLAT ist dann mit jedem internetfähigen Rechner über einen Webbrowser möglich.

2. Poduktübersicht

Folgende Erweiterung stehen nach dem 1.Review fest:

- **Erstellung und Einbindung von (OLAT-globalen) Ressource "Akademisches Jahr"** -(Anzeige von Studententerminen)-Der OLAT-Benutzer bekommt die Möglichkeit sich durch Auswahl dieser Option, alle Termine, die das Akademische Jahr betreffen, anzeigen zu lassen. Die Termine des akademischen Jahres sollten zu diesem Zweck in einer speziellen Tabelle im Filesystem gespeichert werden, da sie an verschiedenen Universitäten unterschiedlich sind. Da die Entscheidung, ob man sich die Daten anzeigen lassen möchte, aber persönlich ist, kann diese Information nur aus dem persönlich Home in der Applikationsschicht stammen. Man muss also während der Implementation von dort aus über die Core Service Schicht auf die Integrationsschicht, in der sich die Daten befinden, zugreifen.
- **Erstellung und Einbindung von Ressource "Feiertage"**-(Anzeige von Feiertag)-Auf Wunsch des Nutzers können Feiertage im Kalender automatisch markiert und als solche erkennbar gemacht werden. Diese sollten aus einer freien Datenbank stammen die dem Nutzer auch die Möglichkeit gibt bundeslandspezifische Pakete von Feiertagen auszuwählen. Die Auflistung, welche Feiertage in welchen Bundesländern arbeitsfrei sind, können in einer Tabelle statisch zu den OLAT-Daten hinzugefügt werden, die nötigen Informationen findet man unter http://www.feiertage.net/bundeslaender.php. Die genauen Daten der Feiertage sind unter http://www.feiertage.net/uebersicht.php?year=2008, beziehungsweise mit der änderung year=2009, ..., year=2037 auslesbar. Auch diese können in einer Tabelle dem Filesystem in der intagration layer hinzugefügt und über eine entsprechende Funktion aus der core service schicht in den persönlichen Kalender in der application layer eingebunden werden.

- **Einbindung einer Ressource "Geburtstage" als Erweiterung der bestehenden Ressource "privater Bereich"** -Auf den Visitenkarten der OLAT-Benutzer, die ihr Geburtstagsdatum frei gegeben haben, soll es eine Schaltfläche geben, die das Geburtstagsdatum des jeweiligen Benutzers zum eigenen Kalender hinzufügt. Dabei soll die core service layer eine Funktion zur Verfügung stellen, die eine Verbindung zwischen dem persönlichen home eines Nutzers, in dem sich sein Profil und seine Visitenkarte befinden, und dem persönlichen Kalender eines anderen Benutzers herstellt.

- **Einbindung einer Ressource "Kurstermine" als Kursbaustein oder Lernressource einschließlich der Integration einer solchen in einen Kurs über die bestehende Schnittstelle "Baustein zu Kurs hinzufügen und Zugang festlegen"**- (Kursbaustein, um

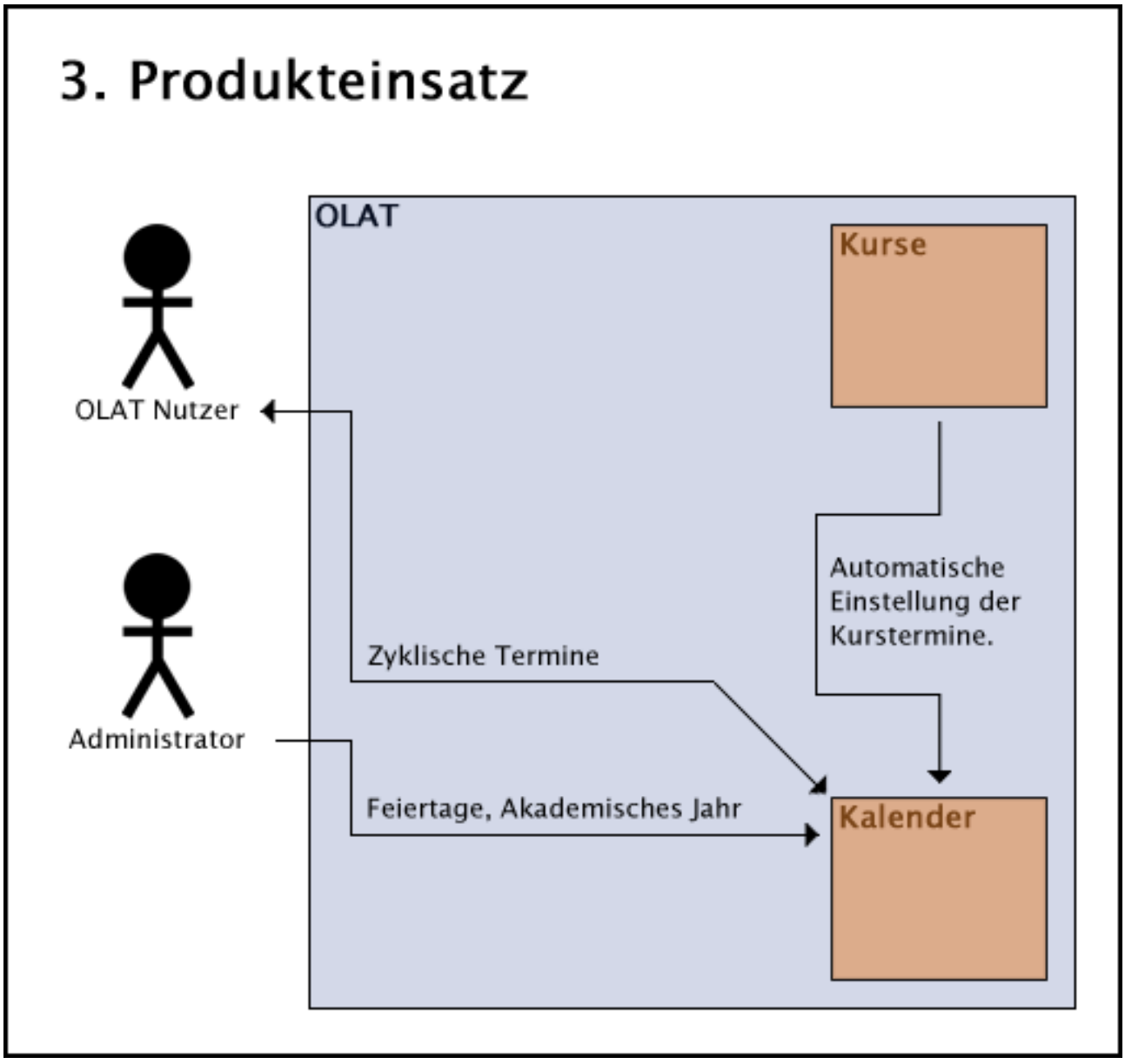
termine automatisch in den Kalender zu integrieren.)-Bei der Kurserstellung bekommt der Erstellende eine Option, die es ihm Ermöglicht zyklische, kursrelevante Termine einzutragen. Dabei wird zwischen optionalen- und obligatorischen Veranstaltungen unterschieden. Die Kursterminintegration ist als neuer Kursbaustein angedacht. In diesem Zusammenhang wäre es sinnvoll, im course edit mode unter course elements die neue Ressource Kurstemine hinzuzufügen. In der core service layer kann eine Funktion die Termine und den Kurskalender inhaltlich verbinden.

Anwendungsbereich- Verwaltung von Terminen der OLAT-Benutzer

Zielgruppen- Nutzer von OLAT Zielgruppe des Endproduktes sind Studenten, Dozenten und deren Gehilfen sowie Professoren

Betriebsbedingungen- Heimrechner, Browser. Die Anwendung soll auf jedem internetfähigen Rechner über einen beliebigen Browser lauffähig sein. Des weiteren muss auf die ständige verfügbarkeit des Produktes geachtet werden.

Mit dem folgendem Diagramm soll eine Ubersicht für die, teilweise hierarchisch angeordneten, Funktionen des OLAT-Kalender gegeben werden. Eine schematische Übersicht der wichtigsten Systemfunktionen:



3. Grundsätzliche Struktur und Entwurfsprinzipien für das Gesamtsystem

OLAT ist primär eine Servlet-Anwendung, die einen Servlet-Container (z. B. Apache Tomcat benötigt, um zu laufen. Sie verwendet zahlreiche Open Source-Frameworks, um die benötigte Funktionalität umzusetzen. Die bekanntesten sind:
- hibernate für das objektrelationale Mapping
- velocity für den Layout-Prozess
- spring für die Konfigurationsverwaltung

Das GUI-Framework ist eine Eigenentwicklung, die komponentenbasiert ist und dem Model-View-Controller-Paradigma folgt. Dabei steht die Wiederverwendbarkeit von Arbeitsabläufen und Komponenten im Vordergrund. Es ähnelt den Java Server Faces, bietet aber umfangreichere Möglichkeiten. Die GUI-Programmierung ähnelt der mit Swing oder SWT. Es gibt Panels, Container mit Layouts, vorgefertigte Komponenten wie Links, Formulare, Tabellen usw. Die Arbeitsabläufe sind in in wiederverwendbare Controller-Klassen gruppiert, so dass bspw. die Nutzersuche überall verwendet werden kann, wo sie benötigt wird. Das Framework basiert auf UTF-8, so dass OLAT problemlos in beliebige Sprachen übersetzt werden kann.

Komponenten- Eine Komponente (Component) ist eine visuelle Repräsentation beispielsweise eines Links, eines Formulars, einer Tabelle usw. Ein oder mehrere Controller können Ereignisse (Events) abfangen, die emittiert werden, wenn der Nutzer z. B. ein Formular abschickt. Jede Komponente muss einen HTML-Renderer (methode getHTMLRendererSignleton (Interface ComponentRenderer)) bereitstellen, der weiß, wie die jeweilige Komponente in HTML repräsentiert wird.

Trennung von Layout und Funktionalität- Geschäftslogik wird in Controller-Klassen (und den von ihnen verwendeten Manager-Klassen) gekapselt, während das Layout im wesentlichen von den verwendeten Cascading Style Sheets beeinflusst wird. Nahezu alle Bilder können auch mittels CSS konfiguriert werden.

Ereignisverwarbeitung- Das GUI-Framework hat im wesentlichen zwei Sachen zu tun, wenn der Nutzer mit der OLAT-Benutzeroberfläche z. B. durch Mausklicks interagiert.

1. Die Anfrage des Benutzers zur betreffenden Komponente befördern
   - Die Komponente im Komponentenbaum der ContentPane des Fensters finden (mit der component-id im URI)
   - die Methode dispatchRequest(userRequest ureq) aufrufen, so dass die Komponente ihren Zustand anpassen kann
   - die Komponente sendet jetzt ein Ereignis eventuell vorhandene Controller (z. B. „Treenode-clicked“ mit der id des angeklickten Knotens).
   - Die Controller, die das Ereignis empfangen, nehmen von der Geschäftslogik definierte Aktionen vor.
   - Der Controller kann jetzt andere Komponenten verändern (z. B. in einem Assistenten einen Schritt weitergehen) und selber Ereignisse an „Eltern“-Controller übergeben (z. B. ein UserChosenEvent).
2. Den neuen Zustand in HTML ausgeben oder, falls es sich um ein PDF-Dokument o. ä. handelt, dieses an den Browser ausliefern.

Das Rendering geschieht, indem die HTML-Ausgabe in einen StringBuffer geschrieben wird, während der Komponentenbaum durchlaufen wird. Am Ende wird das ganze StringBuffer-Objekt zum Client (Webbrowser) geschickt. Das benötigt zwar Speicher, hat aber den Vorteil, dass der Benutzer niemals eine halb gerenderte Seite sieht. Die Seite wird entweder korrekt funktionieren, oder eine vernünfige Fehlermeldung mit Informationen zur Fehlerbehebung für

den Support zeigen.
3. OLAT und AJAX
In Version 5.2.0 wurden erstmals AJAX-Funktionen in OLAT integriert. Die Komponentenarchitektur erlaubte es, einen Modus einzuführen, in dem einzelne Komponenten (im Gegensatz zum gesamten Fenster) neu geladen werden können. Die Komponenten werden dann über das Document Object Model (DOM) in das bestehende Fenster eingebaut. Daher ist es für Entwickler sehr einfach, die AJAX-Techniken einzusetzen. Um Probleme zu vermeiden, wird der AJAX-Modus nur in vollständig getesteten Webbrowsern aktiviert, nicht getestete Browser erhalten bei jedem Request das komplett neue GUI-Layout.

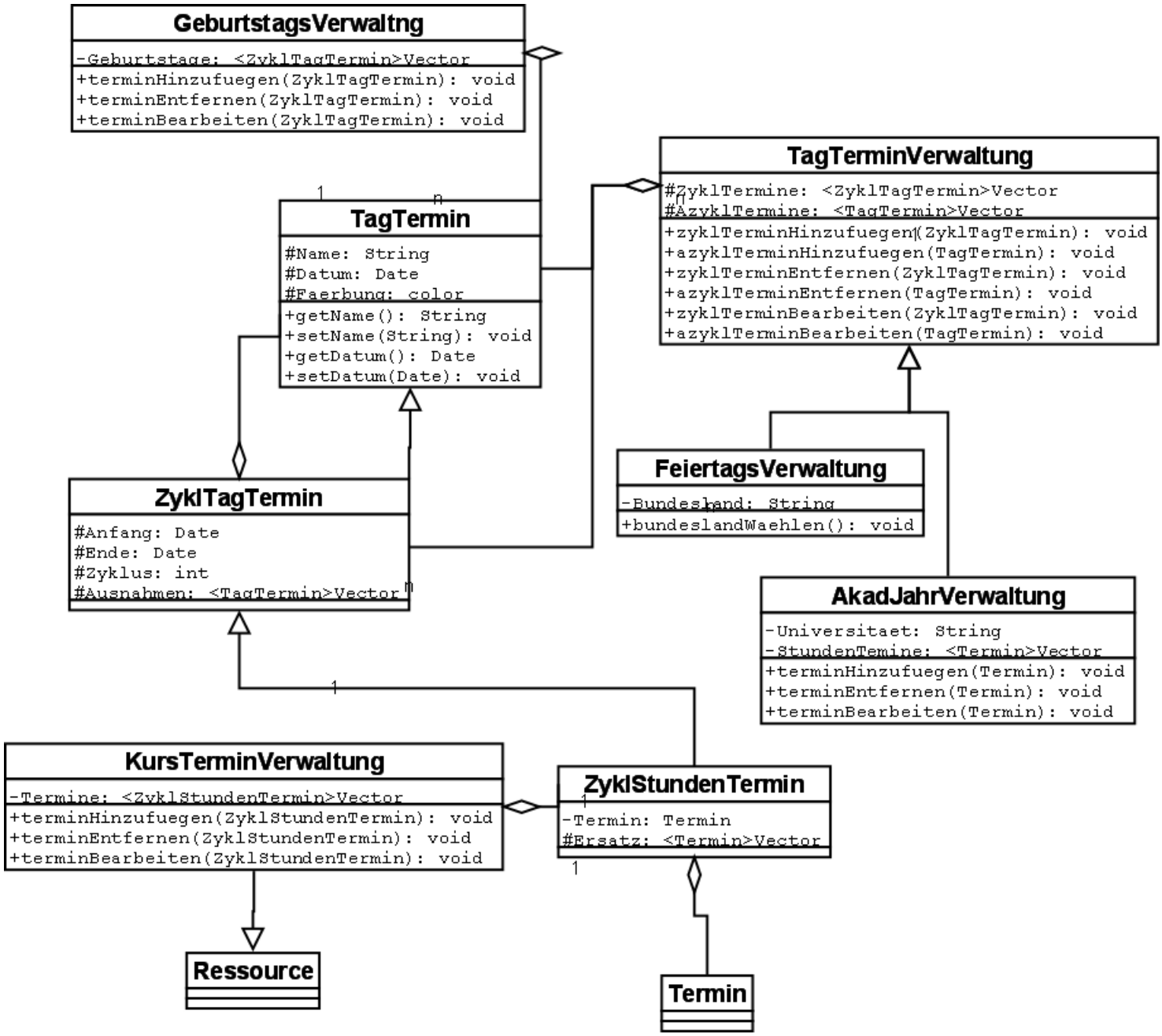
Features, die AJAX benötigen- Der AJAX-Modus ermöglicht es, den Server zu „pollen“, also regelmäßig Anfragen an den Server zu stellen, um Änderungen an Komponenten übermitteln zu können. Das wird beispielsweise im InstantMessagingMainController verwendet, um empfangene Nachrichten vom Server zu holen.

Verarbeitung von externen Ereignissen im Controller- Eine Komponente kann Mausklicks, Formulare usw. verarbeiten, die der Nutzer an OLAT schickt. Gelegentlich möchte man aber auch andere Ereignisse verarbeiten, beispielsweise eine XMPP-(Jabber-)Nachricht von einem anderen Server. Für diesen Zweck kann man das Interface GenericEventListener implementieren. Auch hierfür ist der InstantMessagingMainController ein Beispiel.

Debugging-Features in OLAT- OLAT hilft dem Entwickler mit einigen eingebauten Features.
- Der „GUI debug Mode“. Oftmals haben neue Entwickler Probleme, zu verstehen, welche Komponente oder welcher Container wo und wann angezeigt wird. Anders herum möchte man oft wissen, welche Komponente für ein bestimmtes GUI-Element verantwortlich ist. Diese Informationen (und weitere) werden im GUI Debug Mode angezeigt.
- Die Velocity-Engine kann Files automatisch aktualisieren, wenn diese verändert wurden.

4. Grundsätzliche Struktur und Entwurfsprinzipien der einzelnen Paket

UML – Diagramm



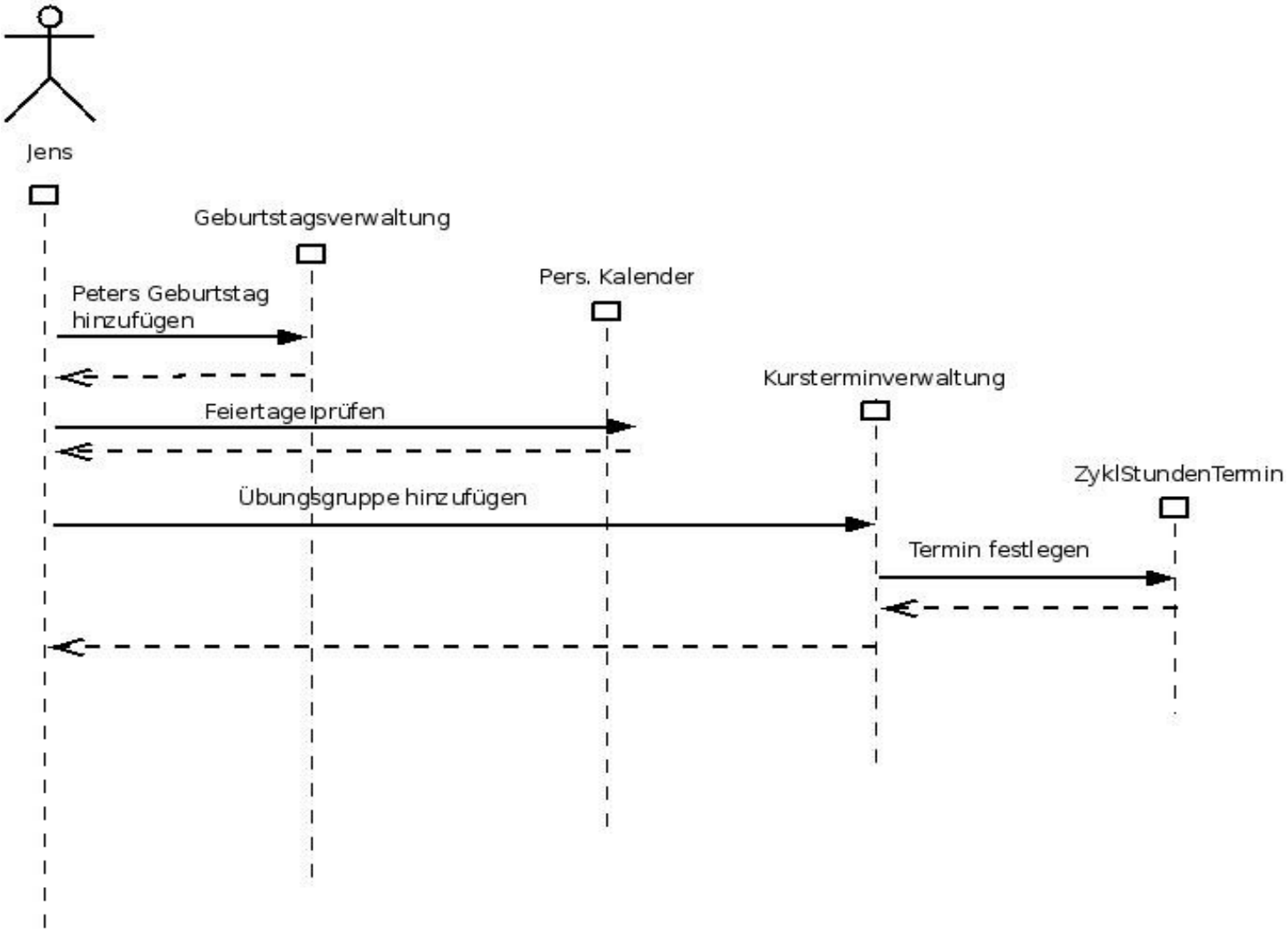
In der Klasse **TagTermin** werden generelle Eigenschaften eines Termins festgelegt, der einen ganzen Tag beschreibt, wie zum Beispiel: 25.12.2008 „1.Weihnachtsfeiertag“ wird rosa eingefärbt. Diese Eigenschaften haben sowohl zyklische als auch nichtzyklische Termine, weshalb es nahe liegt, dass die zyklischen Tagtermine davon erben.
In dieser Klasse werden die Methoden getName und setName aufgeführt. Das soll verallgemeinernd dafür stehen, dass bei allen Klassen die nötigen Methoden zum setzen und auslesen der privaten Variablen implementiert werden. Der Einfachheit und der Übersicht halber haben wir darauf verzichtet, diese Art Methoden in jeder Klasse anzuführen.

Die Klasse **ZyklTagTermin** erbt alle Eigenschaften von TagTermin und stellt alle zyklischen Termine, die sich am bestimmten Tag ohne Zeitangabe (wie z.B. Geburtstage) wiederholen dar. Dazu hat er einen Anfang und ein Ende vom Typ date. Der Anfang wird in der Variable Datum, aus der Mutterklasse TagTermin gespeichert. Wenn kein Datum in einer der Variablen steht, bedeutet dies, dass der Termin schon oder für immer in seinem Zyklus stattfindet. Der Zykus ist vom Typ Integer, darin ist die Periode des Termins codiert, 1 bedeutet täglich, 7 wöchentlich, 14 14-tägig, 31 monatlich, 365 jährlich. Es gilt zu beachten, dass es sich dabei nicht um ein konkretes Inkrement handelt, sondern eben nur um eine Kodierung, weil Monate und Jahre variable Längen haben können.
Im Vektor Ausnahmen ist Platz für eine Reihe von TagTerminen, die Unregelmäßigkeiten repräsentieren. Zum Beispiel wird der tägliche Termin „vorlesungsfreie Zeit“ zum Jahreswechsel durch den Feiertag „Neujahr“ unterbrochen, dann soll dem Kalenderbenutzer ausschließlich der Feiertag angezeigt werden.
Es wäre natürlich auch möglich gewesen, einen zyklischen Termin als Vektor von einfachen Terminen darzustellen, aber das würde unnötig viel Speicherplatz fordern.

Die Klasse **ZyklStundenTermin** erbt alle Eigenschaften von ZyklTagTermin und erweitert die Klasse um eine konkrete Zeitangabe, wie sie Termine im OLAT bereits haben. Hier werden Termine wie „Vorlesung ADS“ jeden Montag 13:15 Uhr gespeichert. Durch den Ersatzvektor kaönnen hier nicht nur Daten, die zum Auslöschen gedacht sind, wie in ZyklTagTermin, enthalten, sondern auch Ersatztermine. Fällt zum Beispiel eine Vorlesung auf einen Feiertag, kann sie am Dienstag darauf um 15:15 Uhr nachgeholt werden.

Verschiedene „Verwaltungsklassen“ sollen entsprechend ihrer Spezifikation Möglichkeiten bieten zyklische und nicht zyklische Termine einzufügen, zu löschen, zu bearbeiten und zu verwalten. Bspw. die Klasse **GeburtstagsVerwaltung** kann einenen zyklischen Termin „Geburtstag“ einfügen, löschen und bearbeiten.
Kursterminverwaltung ist selbst ein Kursbaustein und erbt daher von einer im System vorhandenen Superklasse (hier **Ressource** genannt). Die Klasse Kursterminverwaltung verwaltet alle zum gegebenen Kurs zyklischen Termine.
Tagterminverwaltung ist eine Klasse, die sowohl zyklische, als auch azyklische Termine verwaltet. Von dieser erben die Klassen **FeiertagsVerwaltug** (bundeslandabhängig) und **AkadjahrVerwaltung** (siehe Glossar), die sich nur dadurch voneinander unterscheiden, dass die Feiertage vom Bundesland, die Daten des Akademischen Jahres aber schon von der Universität abhängen. Und dadurch, dass es Termine im Akademischen Jahr gibt, die auch eine Uhrzeit haben, zum Beispiel: „Feierliche Immatrikulation“.

Sequenzdiagramm



Jens ist bereits im Olat eingeloggt und sieht sich gerade die Visitenkarte von seinem Freund Peter an. er fügt den Geburtstag von Peter in seinen Kalender ein. anschließend prüft er im Kalender, ob an Petersgeburtstag oder am tag darauf ein Feiertag ist, oder vielleicht vorlesungsfreie zeit, damit die freunde entscheiden können, ob es besser ist reinzufeiern oder lieber am Geburtstag selbst. Jens ist studentische Hilfskraft an der uni und leitet eine von mehreren, die Vorlesung "Logik" begleitenden, Übungsgruppen. nach Abgleich mit seinem stundenplan, fügt er die Übungsgruppe 2 montags 11:15 A-Woche zu den anderen Kursterminen hinzu.