Tele-TASK-Portal – Notizsystem

Erste Ausarbeitung

Seminar: „Konzepte und Methoden der Webprogrammierung“
Prof. Dr. sc. nat. Christoph Meinel und Dipl.-Informatiker Matthias Kutzner

WS 05/06 am Hasso-Plattner-Institut für Softwaresystemtechnik

beteiligte Studenten:

Thomas Janda (thomas.janda@hpi.uni-potsdam.de)

Dominic Letz (dominic.letz@hpi.hpi.uni-potsdam.de)

Jonas Trümper (jonas.truemper@hpi.uni-potsdam.de)

Sebastian Voigt (sebastian.voigt@hpi.uni-potsdam.de)

Inhaltsverzeichnis

1 Einleitung

1.1 Rahmenbedingungen:

Unter Verwendung von PHP, MySQL, Javascript und der AJAX-Technologie soll eine interaktive Verwaltung von Notizen innerhalb eines neuen tele-TASK-Portal-Systems erstellt werden. Das tele-TASK-Portal stellt verschiedene nationale und internationale Veranstaltungsreihen zur Verfügung. Die zu erstellende Applikation soll sich in ein in der Entwicklung befindliches Komponentenframework einbetten. Insbesondere sind definierte Schnittstellen zu einer Internationalisierungs- , Navigations- und Benutzerkomponente zu nutzen.

1.2 Anwendungsfälle:

- Benutzer des tele-TASK-Portals erhalten die Möglichkeit während einer laufenden Vorlesung beliebige eigene Notizen innerhalb des Veranstaltungskontextes des tele-TASK-Portals abzulegen. Diese Notizen kann ein Benutzer später an dieser Stelle wieder abrufen oder bearbeiten.
- Vortragende deren Veranstaltungen durch das tele-TASK-Portal veröffentlicht sind, können für die Veranstaltung relevante Inhalte, Ankündigungen oder Anmerkungen innerhalb des Veranstaltungskontextes des tele-TASK-Portals anbieten.
- Benutzer haben die Möglichkeit sich in Gruppen zu organisieren, die einen veranstaltungsübergreifenden Kontext (Vorlesungsreihen) haben, in dem sie gemeinsame Notizen festhalten können.

2 Konzept / eigene Anforderungen

2.1 Funktionale Anforderungen:

Die Nutzer der tele-TASK-Webseite sollen die Möglichkeit erhalten Notizen zu einzelnen Vorlesungen anzulegen. Jede Notiz erhält dabei einen Zeitstempel, der sie eindeutig einem Zeitpunkt innerhalb der Multimediapräsentation zuordnet. Wird beim Abspielen dieser Zeitpunkt erreicht, lädt das Notizsystem die Notiz und präsentiert diese dem Nutzer.

Für den Einsatz innerhalb des tele-TASK-Portals sind zwei Modi vorgesehen, der Einzelnutzermodus und der Gruppenmodus.

Einzelnutzermodus

Der Einzelnutzermodus bietet einem einzelnen Nutzer die Möglichkeit private Notizen anzulegen und diese zu verändern.

Gruppenmodus

Der Gruppenmodus soll das Anlegen von Notizen ermöglichen, die von mehreren Nutzern gelesen und bearbeitet werden können. Eine Gruppe wird immer genau einer Vorlesungsreihe zugeordnet sein.

Um eine optimale Bedienung zu ermöglichen, soll ein schneller und unkomplizierter Wechsel zwischen den beiden Modi möglich sein. Dazu wird die Gruppenfunktionalität von der globalen Benutzerverwaltung losgelöst sein.
So hat der Nutzer die Möglichkeit, während er den Stream betrachtet, sowohl für sich selbst als auch für die Gruppe Notizen zu bearbeiten/ zu erstellen.

Das System wird keine Gruppenverwaltung bieten. Eine neue Gruppe wird angelegt, sobald sich ein Nutzer mit bisher unbekannten Logindaten in eine Gruppe einloggen möchte, meldet er sich mit dem Gruppennamen und dem Passwort einer bereits vorhandenen Gruppe an, so wird er in dieser angemeldet.

Dies hat den Vorteil, dass Nutzer des Portals selbständig neue Gruppen anlegen oder auch neue Mitglieder in ihre vorhandene Gruppe integrieren können.

Zusätzlich soll es möglich sein, öffentliche Notizen anzulegen. Es sollen sowohl der Dozent als auch Studenten (diese nur nach Überprüfung der Notiz durch einen Redakteur) die Möglichkeit haben Notizen zu veröffentlichen. Solche Notizen können anschliessend von allen eingelogten Nutzern gelesen werden, die sich den betreffenden Stream ansehen.

Alle Notizen sollen unmittelbar ab dem Zeitpunkt der Erstellung allen berechtigten Nutzern zur Verfügung stehen. Bereits angemeldete Benutzer sollen also auch Notizen präsentiert bekommen, die geschrieben wurden, nachdem sie den Stream geöffnet haben.

Allen Nutzern soll in der Ansicht eine Liste aller von ihnen erstellten Notizen gezeigt werden. Befinden sie sich im Gruppenmodus, so sollen ihnen auch, falls vorhanden, die Gruppennotizen angezeigt werden. Im Einzelnutzermodus bekommen sie die von den Redakteuren für die Öffentlichkeit freigeschalteten Notizen präsentiert.

Wie schon erwähnt orientiert sich die Anzeige der Notizen an Hand der Position innerhalb des Präsentationsmediums, das dem Nutzer parallel zur Notizfunktion angezeigt wird. Der Übersichtlichkeit halber findet eine Untergliederung der Notizen in private und öffentliche, bzw. Gruppennotizen statt, die unabhängig voneinander angezeigt werden.

Auf Wunsch soll der Nutzer sich auch bestimmte Notizen anzeigen lassen können, indem er diese direkt per Maus selektiert.
Um hierbei den Bezug der Notiz zur Vorlesung herzustellen, soll das Medium an die Stelle, zu welcher die Notiz abgespeichert wurde, springen.

Da die Notizen beim Erreichen der betreffenden Streamposition geladen werden sollen, müssen diese einen gewissen zeitlichen Abstand zueinander besitzen, da sonst Notizen, die zur gleichen

Zeit gespeichert wurden, oder einen geringen zeitlichen Abstand zueinander besitzen, dem Nutzer gar nicht oder zu kurz angezeigt werden.

Daher sollen solche Notizen innerhalb einer Notizdatums zusammengefasst werden, und somit als eine Notiz abgespeichert und angezeigt werden, wobei aber noch eine Trennung der Notiztexte erfolgen muss um dem Nutzer anzuzeigen, dass es sich tatsächlich um mehr als nur eine Notiz handelt, was vor allem bei zusammengefassten öffentlichen Notizen von Interesse ist.

2.2 Nicht-funktionale Anforderungen:

Prinzipiell lässt sich die Logik des Notizsystems auf 2 Bereiche aufteilen:

- Server (Speicherung und Verwaltung der Daten sowie deren Aufbereitung)
- Client (zeigt die Daten an und stellt ggf. weitere Anfragen an den Server)

Normalerweise lassen sich die Inhalte einer Seite, die ein HTTP-Client bzw. Browser vom Server geladen hat nur durch ein Neuladen der Seite aktualisieren. Für unsere Anwendung jedoch ist dieses Szenario absolut undenkbar, denn würde die Seite neu geladen, so würde auch der Stream von vorne beginnen und der Nutzen einer Notizfunktion wäre
ad absurdum geführt.

Hier muss also eine andere Lösung her, die es erlaubt nur einen Teil der Seite zu aktualisieren bzw. auszutauschen, also mit aktuellerem Inhalt zu füllen.
Auf diesen Aspekt gehen wir unter *„eingesetzte Technologien"* genauer ein.

Eine Notiz muss zur nächsten Notiz mindestens einen Abstand von 5 Sekunden zur vorigen haben.

Zusätzlich benötigt unsere Notizfunktion natürlich Tabellen in einer Datenbank, in der die Notizen und dazugehörige Daten abgelegt werden können.

3 Eingesetzte Technologien:

Die wichtigsten Technologien, welche im tele-TASK-Portal Verwendung finden sind:

PHP mit Smarty, eine Datenbank (MySQL) und Ajax (Javascript und XMLHttpRequest)

3.1 PHP:

PHP ist eine Open-Source Skriptsprache, welche bevorzugt zum Schreiben von Webanwendungen Verwendung findet und unter der 'PHP License' vertrieben wird. Der Interpreter läuft dabei auf dem Webserver. Es muss also kein Code zum Client übertragen werden. Bloß der jeweilige Output, meist ein HTML-Dokument, wird an den Browser geschickt. Die Skripspracheneigenschaft der immer wiederkehrenden Neuinterpretation beim jeweiligen Aufruf von PHP-Skripten schlägt sich bei größeren Projekten gravierend in der Perfomance nieder, welche quasi in den Keller geht. Abhilfe kann hier ein Vorcaching bieten, was aber nicht standardmäßig vorgesehen ist.

Ansonsten bietet die Sprache vielerlei Funktionalitäten, wie z.B. Objektorientierte Programmierung, leichte Datenbankanbindungen und einer großer Verbreitung/Akzeptanz unter den Webentwicklern. PHP ist der Hauptprogrammiersprache im TeleTASK-Projekt, was auch in den kommenden Versionen so vorgesehen ist.

3.2 Smarty:

Smarty ist eine Template Engine, welche in PHP geschrieben ist (eine PHP-Bibliothek). Sie findet im TeleTASK-Projekt Verwendung um den eigentlichen Code von dessen Ausgabe zu trennen, wofür Smarty prädestiniert ist. Somit wird eine bessere Strukturierung gewährleistet, was wiederum mit einer leichteren Wartun/Veränderung einhergeht, z.b. Layoutanpassungen. Smarty liegt unter der Lesser General Public License (LGPL) und kann demzufolge frei genutzt, verändert und angepasst werden. Ein weiterer Vorteil zur Performancesteigerung bringt Smarty gleich mit, nämlich eine Art Compiler, der die Smarty-Template Ausgaben in einem Cache speichert, wodurch die eigentliche Ausgabe stark beschleunigt wird.

3.3 MySQL:

MySQL ist ein freies Datenbankverwaltungssystem (RDBMS) und läuft auf vielen verschiedenen Betriebssystemen. Es handelt sich um ein relationales Datenbanksystem, welches sich großer Beliebtheit erfreut und im Zusammenhang mit der in PHP geschrieben Software phpmyadmin sehr leicht verwalten lässt. Ab der Version 5 gibt es sogar Stored Procedures- und Trigger-Unterstützung. Zur Zeit läuft auf dem TeleTASK-Server ein MySQL-Server in der Version 4.1.

3.4 Ajax:

Akronym für: Asynchronous Javascript and XML

Ajax ist ein Konzept der Übertragung zwischen Server und Client (Browser). Die Haupteigenschaft besteht darin, dass bei einer Anfrage nicht die komplette HTML-Seite neu geladen werden muss. Es werden nur die benötigten Daten (asynchron) nachgeladen, so dass die anderen Bereiche bestehen bleiben und in der Zeit weiter benutzt werden können. Ajax ist keine neue Erfindung. Es vereint lediglich bekannte Technologien unter einer Haube mit dem Ziel Webanwendungen so aussehen zu lassen als handele es sich um eine Desktopanwendung.

Ajax vereint folgende Technologien (die wichtigstens):

- HTML, DOM und CSS
- Javascript
- XMLHttpRequest-Objekt

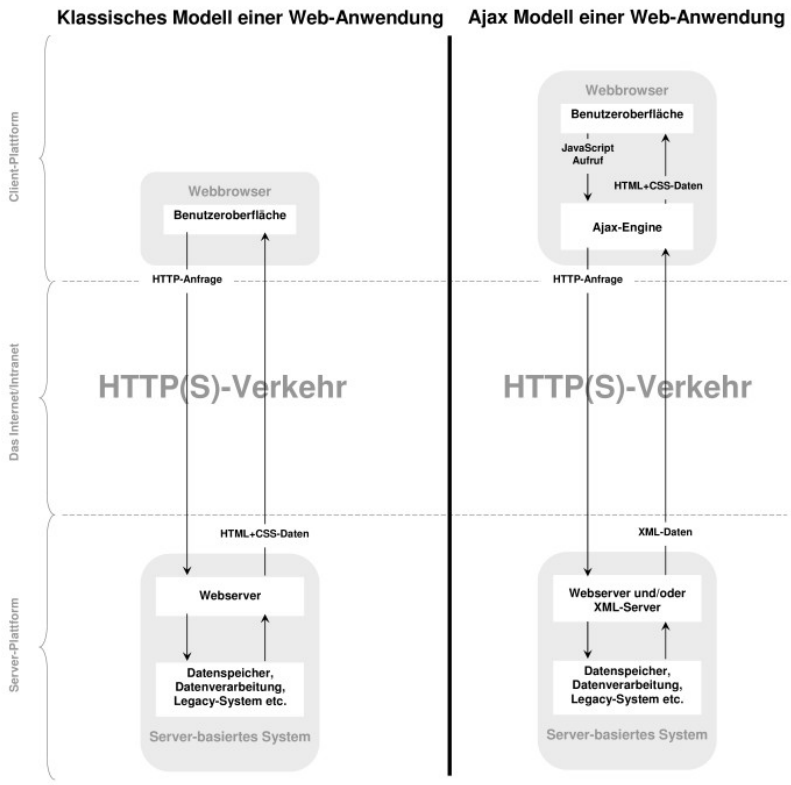

Abbildung 1: Ajax Veranschaulichung - Quelle: wikipedia.de

3.4.1 OpenRico:

OpenRico ist eine offene Javascript-Bibliothek, mit deren Hilfe es möglich ist größere Webapplikationen mit Grundfunktionen auszustatten, welche sich abgekapselt in der rico.js befinden. Somit ist es für den Anwender leichter Anwendungen zu schreiben, denn er muss sich um diverse Sachen (z.b. Runden von Kanten) nicht mehr kümmern, sondern sie einfach nur noch nutzen. Openrico bietet vollständige Ajaxunterstützung. Ebenfalls erweitert um z.B. drag & drop Funktionen. Die Bibliothek besteht aus Klassen (objektorientiert) und benutzt das weit verbreitete Prototype-Pattern. OpenRico steht unter der Apache-Lizenz und kann demnach in jedem Umfeld eingesetzt bzw. modifiziert werden.

Im Endeffekt hat der OpenRico Nutzer nur Vorteile und kann die Bibliothek dazu verwenden, um Entwicklungsarbeit einzusparen.

Veranschaulichung des einfacheren Umgangs mit z.B. Ajax-Requests: Register- und Send-Request:

```
ajaxEngine.registerRequest('getNotiz', 'ajax.php');
.
.
ajaxEngine.registerAjaxElement( 'notiz_privat' );
.
.
function getNotiz(selectBox)
{
  var params = selectBox.value.split(',');
  var id     = params[0];
  var start  = params[1];

  ajaxEngine.sendRequest( 'getNotiz', 'id = ' + id );
}
```

Man benutzt einfach nur die schon vorhandenen Methoden der ajaxEngine zur Registrierung und zum Senden. Die Identifizierung erfolgt über die ID.

3.4.2 XMLHttpRequest

XMLHttpRequest ist die entscheidende Technik hinter Ajax, mit deren Hilfe es möglich ist Webanwendungen wie Desktopanwendungen aussehen zu lassen. Dieser Eindruck wird erst möglich, indem Seiten bei einer Aktion nicht komplett neugeladen werden müssen, sondern nur Teile davon. Dieses Nachladen erfolgt wiederum asynchron. Demzufolge kann ein Script weiter abgearbeitet werden, ohne auf die Antworten der Anfrage warten zu müssen. Die Anfrage liefert die Daten im XML Format zurück.

4 Umsetzung

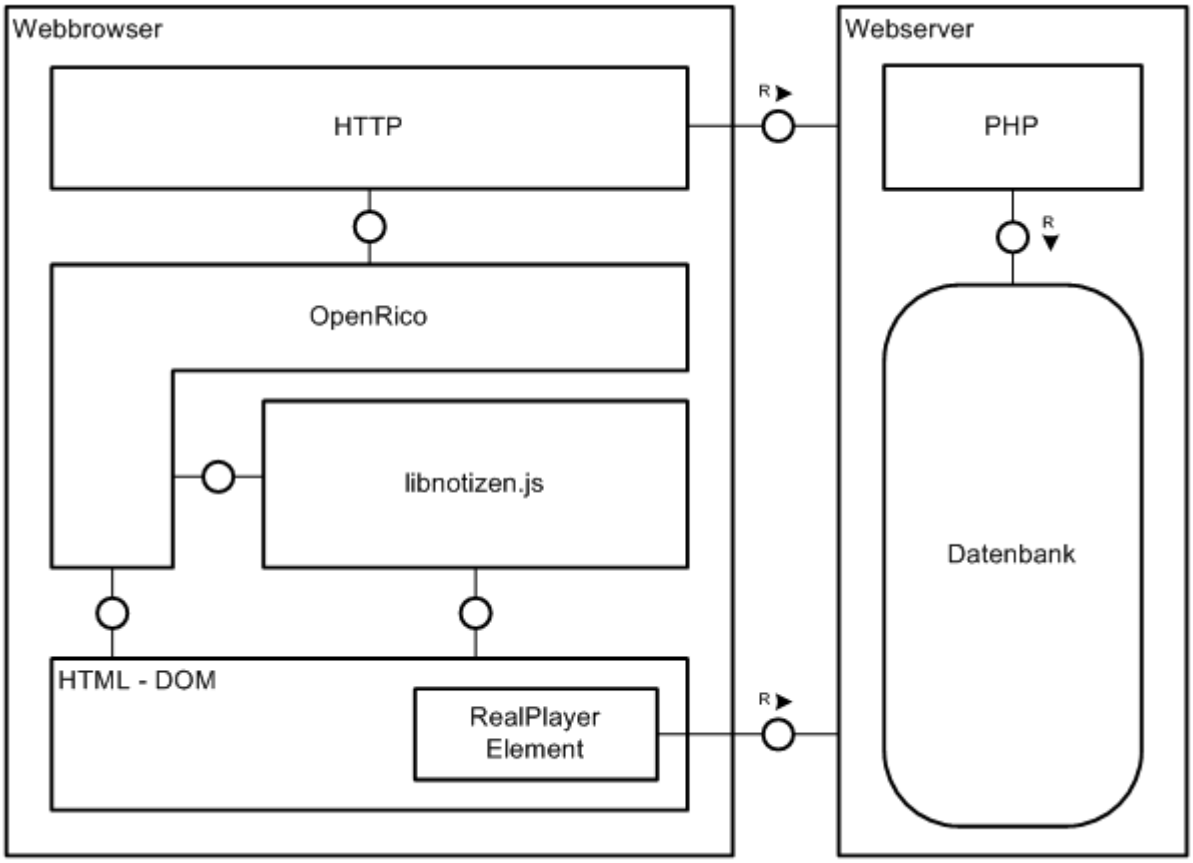

Abbildung 2: Aufbau "Von unten nach oben"

4.1 Überblick

Für einen Benutzer der Anwendung stellt der Webbrowser des Benutzers die Anwendungsoberfläche dar. Über den Browser interagiert der Benutzer mit den HTML-Elementen. Diese HTML-Elemente interagieren wiederum mit Javascript, welches über das OpenRico-AJAX-Framework zugriff auf HTTP-Kanäle zum Webserver hat. Dort werden die Anfragen von einer PHP-Anwendung verarbeitet. Die Antworten haben das XML-Format und gelangen auf gleichem Weg wieder zurück und werden über das OpenRico-AJAX-Framework als Ereignisse Rückruf-Funktionen zugänglich gemacht.

4.2 Javascript

Innerhalb der Anwendung existieren drei Schichten, die jede Notizenanfrage des Benutzers durchlaufen muss. Auf der abstraktesten Ebene befindet sich das Javascript *User-Objekt*. Diese enthält alle logischen Funktionen die ein Benutzer innerhalb des Notizensystem vollführen kann:

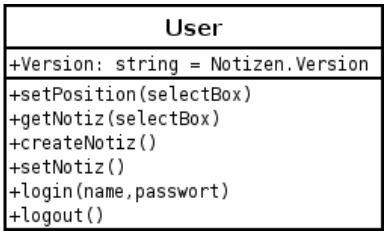

Abbildung 3: JS User-Objekt

- Im Video an die Position einer Notiz springen
- Eine Notiz laden
- Eine neue Notiz erstellen
- Eine vorhandene Notiz bearbeiten
- Sich in eine Gruppe einloggen
- und aus einer Gruppe wieder ausloggen

Jede dieser Aktionen entspricht einem Klick auf eines der Elemente im HTML-Formular der Seite. Das User-Objekt verarbeitet diese Anfragen jedoch nicht selbst sondern stellt vielmehr eine Fassade (Facade-Pattern) für die Funktionen des Notizen-Objektbaumes da. Wird beispielsweise die Methode "setPosition" aufgerufen, wird die entsprechende Methode am Notizen.Realplayer-Objekt ausgeführt. Die Definition des User-Objektes findet sich in der Datei libnotizen.js ab Zeile 268.

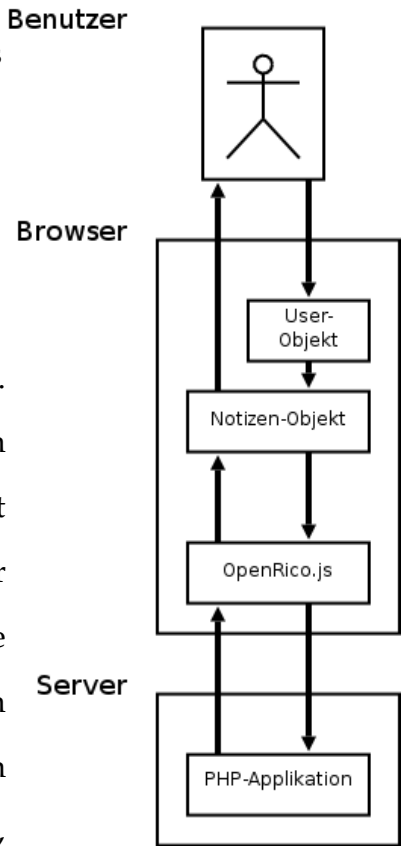

Abbildung 4: Benutzeranfrage "Von oben nach unten"

Die zweite Ebene der Anfragebearbeitung ist das *Notizen-Objekt*. Dieses Objekt enthält mehrere Unterobjekte, weswegen auch von einem Notizen-Objektbaum gesprochen wird. Dieses Objekt nimmt alle durch die Benutzeranfrage erforderlichen Änderungen an der Anwendungsoberfläche vor. So Synchronisiert es z.B. die angezeigte Notiz mit den der Laufleiste des Realplayers. Müssen durch die Benutzeranfrage neue Notizdaten vom Server gelesen oder die Berechtigung zu einem Gruppenzugriff bestätigt werden, leitet das Notizen-Objekt die Anfrage an die dritte Schicht der Bearbeitung das *OpenRico-AJAX-Framework* weiter. Dieses kommuniziert mit der PHP-Anwendung. Sobald die Applikation eine Anfrage bearbeitet hat, werden die Ergebnisse über Rückruf-Funktionen des

Notizen-Objektes verarbeitet, und in der Anwendungoberfläche veröffentlicht. Der gesamte Notizen-Objektbaum findet sich innerhalb der Datei libnotizen.js bis Zeile 268. OpenRico befindet sich in der Datei rico.js zusammen mit prototype.js welches für die Ausführung von OpenRico notwendig ist.

4.3 Notizen-Objektbaum

Die Wurzel des Baumes ist das eigentliche Notizen-Objekt. Es enthält Funktion zum

- Initialisieren
- Aktualisieren der privaten/Gruppen-Notizliste
- Aktualisieren der öffentlichen Notizliste
- Senden eines AJAX-Befehls
- Verarbeiten einer Notizveränderungs bestätigung des Servers
- Verarbeiten einer Authentifizierungsantwort des Servers

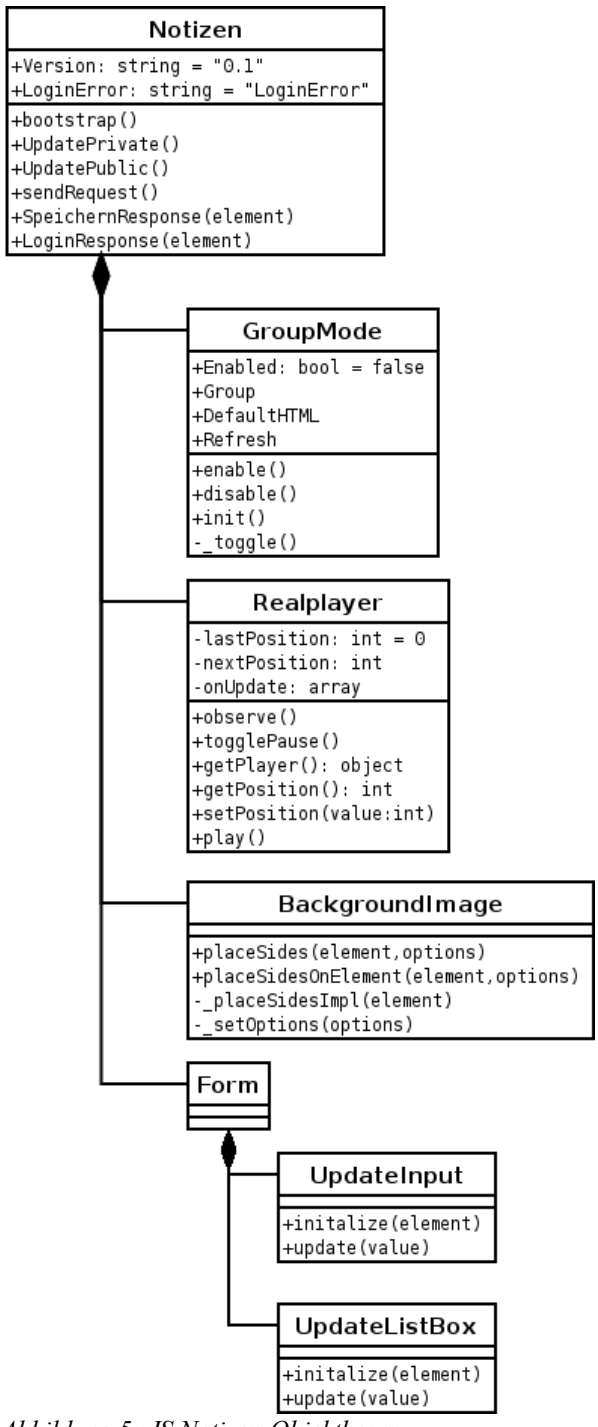

Abbildung 5: JS Notizen-Objektbaum

Dabei hat die globale Initialisierungsfunktion "bootstrap" die Aufgabe die Rückruf-Methoden "UpdatePublic" und "UpdatePrivate" an einem Zeitgeber, die Rückruf- Methoden "SpeichernResponse" und "LoginResponse" am OpenRico-AJAX-Framework, und zwei Rückruf-Funktionsobjekte vom Typ "Form.UpdateListBox" am Realplayer-Objekt zu registrieren Des weiteren bereitet die Initialisierungsfunktion die Awendungsoberfläche vor. Dabei wird auch die "init"-Methode des *GroupMode-Objektes* aufgerufen, welches sich darum kümmert die

Anwendungsoberfläche zwischen Gruppenmodus und Privatnotizenmodus umzuschalten. Es startet die Anwendung zuerst im Privatnotizenmodus.

Das *Realplayer-Objekt* stellt die Schnittstelle zum dargstellten Realplayer-Video-Strom der Anwendung dar. Neben der Möglichkeit den Strom zu Steuern und die derzeitige Position zu ermitteln bietet es einen Benachrichtigungsmechanismus, um Funktionsobjekte regelmäßig über die Veränderung der Streamposition zu informieren.

Das *BackgroundImage-Objekt* stellt HTML-Manipulationsmethoden zur Verfügung, die es bei der Initialisierung ermöglichen dynamisch Hintergrundbilder links und rechts in Titelleisten zu platzieren. Dies wird genutzt um sich in das tele-TASK Design einzubetten, ohne die HTML-Struktur zu komplizieren.

Das *Form-Objekt* enthält zwei Protoyp-Funktionsobjekte, deren Instanzen am Realplayer-Objekt registriert werden können:

- Das *UpdateInput-Funktionsobjekt* aktualisiert das bei Konstruktion übergebene "<input>"-Element regelmäßig mit der derzeitigen Realplayerposition. (Dieses Funktionsobjekt wird nur innerhalb von Entwicklungszyklen verwendet)
- Das *UpdateListBox-Funktionsobjekt* aktualisiert das bei Konstruktion übergebene "<select>"-Element regelmäßig, indem es dafür sorgt, das immer ein Notizeintrag ausgewählt ist, welcher der derzeitigen Realplayerposition entspricht. Bei Bedarf lädt es dazu eine neue Notiz vom Server.

4.4 PHP

Die PHP-Anwendung des Servers bietet zunächst zwei grundlegende Funktionen:

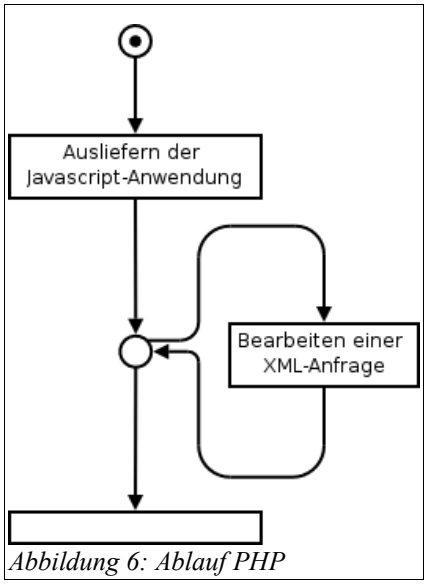

Abbildung 6: Ablauf PHP

- Das ausliefern einer bestimmten Multimediapräsentation (Realplayer) mit eingebetteter Javascript-Anwendung
- Das Beantworten über HTTP geschickter Anfrage der Javascript-Anwendung mit XML

Nach den Anforderungen der Anwendung muss jeder lesbare Text wie z.B. Steuerelementbezeichnungen und Fehlermeldungen in über eine definiert PHP-Schnittstelle entsprechend der Spracheinstellungen des Benutzers übersetzt werden. Um dies mit der Javascript-Anwendung zu ermöglichen werden Teile der Javascript-Anwendung bei der Auslieferung durch die PHP-Anwendung übersetzt. Dies geschieht durch die Verwendung der Singleton-Klasse (Singleton-Pattern) NotizRuntime. Diese Klasse kapselt alle der PHP-Anwendung extern zur Verfügung gestellten Funktionalitäten, wie Benutzer- und Veranstaltungverwaltung, Sessionführung, Datenbankverwaltung und Internationalisierung. Die auszuliefernde Seite wird durch eine Smarty-Klasse gerendert, welche jeweils mit einer Referenz auf die aktuelle Veranstaltung, den Benutzer und die NotizRuntime-Klasse selbst gestaret wird. Innerhalb der Smarty-Vorlage werden nun die zu übersetzenden Ausdrücke, während der Auslieferung ersetzt. (Im Quelltextbeispiel wird auch dargestellt wie das Rendern der Realplayerinstanz an die externe Veranstaltungsverwaltung delegiert wird)

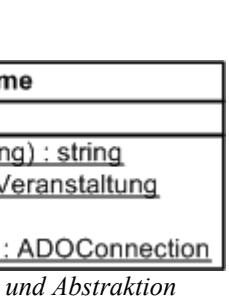

Abbildung 7: Singleton-Klasse und Abstraktion

```
<table>
  <tr>
    <td class="mainLeft">
      <div id="stream">
        {$veranstaltung->getContent()}
      </div>
    </td>
    <td>
        <div id="mainRight">
            <div class="element" id="logout" style="display:none"
                <h1>{$N->translate('Gruppe')}</h1>
                <button type="button" onclick="User.logout();">{$
```
Abbildung 8: Internationalisierung im Smartyquelltext: index.tpl

4.5 Verarbeiten von Anfragen

Unmittelbar nach der Auslieferung erzeugt die Javascript-Anwendung die ersten Anfragen, die von der PHP-Anwendung zu beantworten sind. Dazu besitzt die Anfrageseite genau sechs Funktionen:

- Holen einer Notiz
- Einloggen eines Benutzers
- Setzen von Notizdaten
- Auflisten der Notizen in einer Gruppe
- Auflisten der öffentlichen Notizen
- Auflisten der privaten Notizen

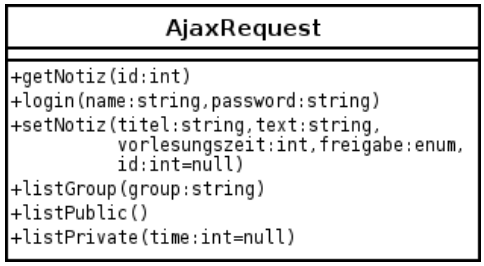

Abbildung 9: Verstandene Anfragen

Alle diese Funktionen arbeiten ohne Angabe von Benutzer- und Vorlesungsinformationen, da diese über die extern zur Verfügung gestellten Funktionalitäten bzw. über die NotizRuntime verfügbar sind. Zu beachten ist am gezeigten UML-Modell, dass es sich hierbei nicht um die Darstellung einer im System vorhandenen Klasse handelt, sondern um die Definition der von der PHP-Anwendung (derzeit über die ajax.php) verarbeitbarer Anfragen. Beim verarbeiten der Anfragen werden haupsächlich die Methoden der Notiz-Klasse benutzt:

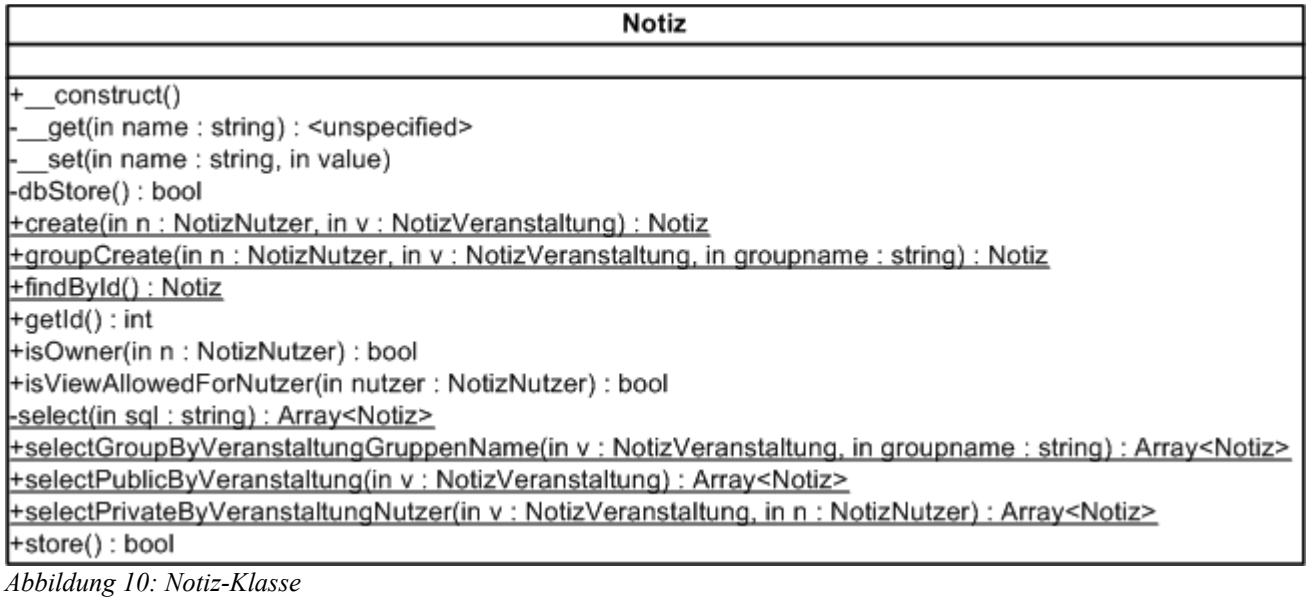

Abbildung 10: Notiz-Klasse

Soll eine Aktion auf einer bestimmten Notiz ausgeführt werden, wird diese zuerst über die statische Klassenmethode "findById" rausgesucht. Bevor eine Notiz an einen Benutzer ausgeliefert wird, wird über die Methode "isViewAllowedForNutzer" geprüft ob der angegebene Benutzer die konkrete Notiz lesen darf. Möchte ein Benutzer eine Notiz schreiben wird dies nur erlaubt wenn die Methode "isOwner" *wahr*

zurückliefert. Dabei ist zu beachten das Gruppennotizen allen Mitgliedern der gleichen Gruppe gehören und somit von allen Gruppenmitgliedern verändert werden können. Erhält ein Benutzer die Erlaubnis eine Notiz zu verändern, wird der Notizdatensatz mittels der impliziten Setzmethode "__set" aktualisiert und durch einen Aufruf auf "store" in die Datenbank geschrieben ("store" prüft zunächst die Datensatzlogik und ruft dann die private Method "dbStore" auf). Durch die Variationen der statischen "select..."-Klassenmethoden werden die verschiedenen "list..."-Anfragen verarbeitet und beantwortet. Dabei wird die private Methode "select" als Vorlagenmethode genutzt (Template Method Pattern). Die Anfrage "login" stellt eine Besonderheit dar. Insofern als dass sie von der NotizNutzer-Klasse verarbeiter wird. Über die statische Klassenmethode "getNutzer" der NotizRuntime (s.o.) wird zunächst ein dem Benutzer entsprechendes Objekt geholt, auf welchem "loginToGroup" mit Gruppenname und Passwort als Parameter ausgeführt wird. Ist der Benutzer nach ausführung Mitglied der Gruppe wird eine Erfolgsmeldung zurückgegeben, andernfalls eine Loginfehlermeldung.

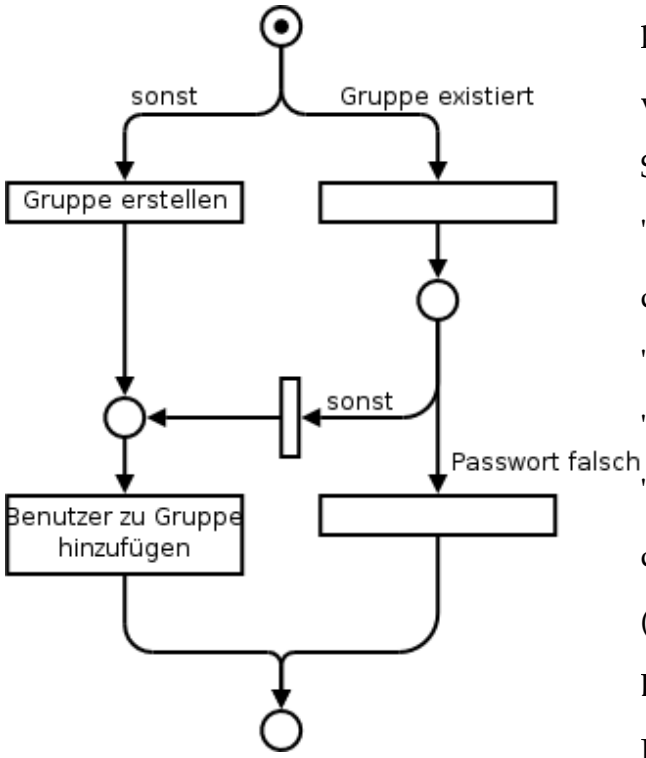

Abbildung 11: NotizNutzer.loginToGroup()

Zur Verwaltung von Nutzern und Veranstaltungen existieren Klassen, die spezifische Funktionen anbieten. So bietet jede Datensatzklasse eine "getId"-Methode über die eine eindeutige Referenzierung eines Datensatzes ermöglicht wird (dargestellt durch die Implementierung der Schnittstelle NotizenDBO). Sowohl die Klasse NotizVeranstaltung als auch NotizNutzer sind innerhalb der vorgestellten PHP-Anwendung

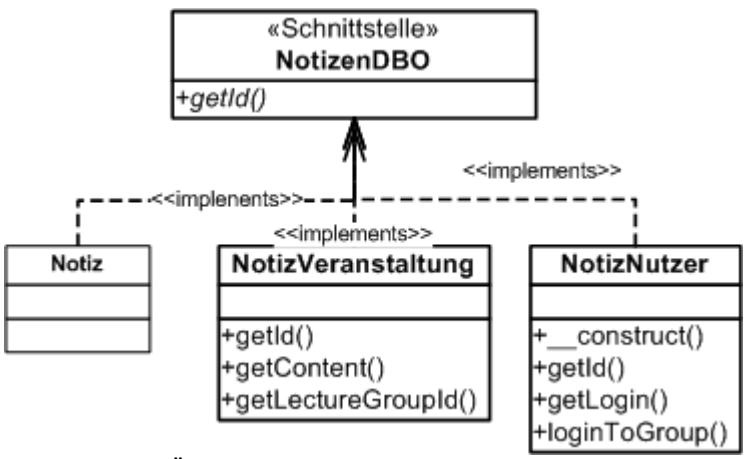

Abbildung 12: Übersicht der Datenklassen

Singleton-Klassen, die sich immer auf den derzeit eingeloggten Benutzer und die derzeit aktuelle Veranstaltung beziehen. Diese Singleton-Objekte werden von der NotizRuntime-Klasse verwaltet. Sie sollen als Adapter für entsprechende Klassen des neuen teleTASK-Frameworks fungieren. Derzeit besteht diese Verbindung jedoch noch nicht.

5 Verwendung

Die Notizfunktion während der Vorlesung erreicht man, indem die Notizfunktion beim Betrachten der Veranstaltung ausgewählt wird. Andere Optionen wären dann noch die Chatfunktion und die reine Betrachtung ohne zusätzliche Aktivität.

Wenn die Option mit Notiz ausgewählt wird, erscheint folgende (ähnliche) Darstellung:

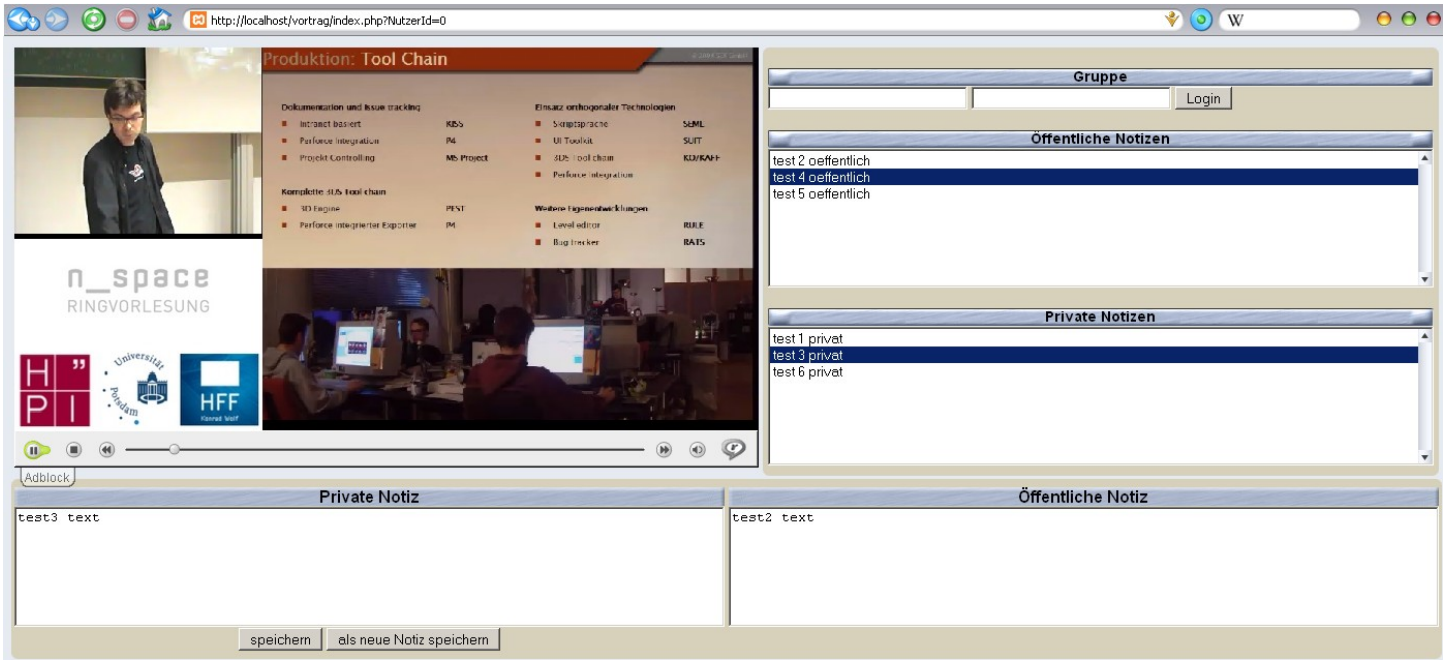

Abbildung 13: Modus: Privatnotiz

5.1 Modus: private Notizen

Für den jeweiligen Nutzer werden seine privaten Notizen, welche vorlesungsgebunden abgespeichert sind (via VorlesungsID), ins private Notizfenster geladen. Falls dieser Nutzer noch keinerlei Notizen selbst verfasst hat, bleibt dieses Fenster natürlich erst einmal leer, solange bis der Nutzer eben eine neue Notiz geschrieben hat, welche dann mit dem jeweiligen Zeitstempel

in der Datenbank abgespeichert wird. Desweiteren befinden sich die zur Vorlesung gehörenden öffentlichen Notizen im Fenster darüber, sortiert nach ihrem Zeitstempel.

Der Nutzer ist nun während der Vorlesungsbetrachtung dazu in der Lage sich Kommentare oder eben nützliche Übersichten als Notiz zu einer bestimmten Zeit abzuspeichern bzw. eigene schon abgespeicherte Notizen zu ergänzen

5.2 Modus: Gruppennotizen

Der Nutzer besitzt die Möglichkeit zur Laufzeit seine Gruppenzugehörigkeit durch einen LogIn zu bestätigen (über den öffentlichen Notizen). Bei korrekter Verifikation durch Gruppenname und dazugehörigem Passwort erscheint im Fenster anstatt der privaten Notizen die Gruppennotizen, welche von verschiedenen Nutzern eingesehen, bearbeitet und erstellt werden können, eben die Gruppenmitglieder.

Jeder Nutzer ist auch in der Lage eine neue Gruppe zu erstellen. Dazu muss einfach nur ein nicht vergebener Gruppenname ins LogIn Feld eingetragen werden, plus Passwort. Daraufhin wird direkt diese neue Gruppe erstellt und neue Mitglieder können, sofern sie das Passwort kennen, sie betreten und Notizen miteinander teilen und verwalten. Ebenfalls ist es möglich wieder in den Modus der privaten Notizen zu wechseln. Beim Klicken auf den Logout-Button wird dies vollzogen. So ist ein zügiges Hin und Her (wem's gefällt) möglich.

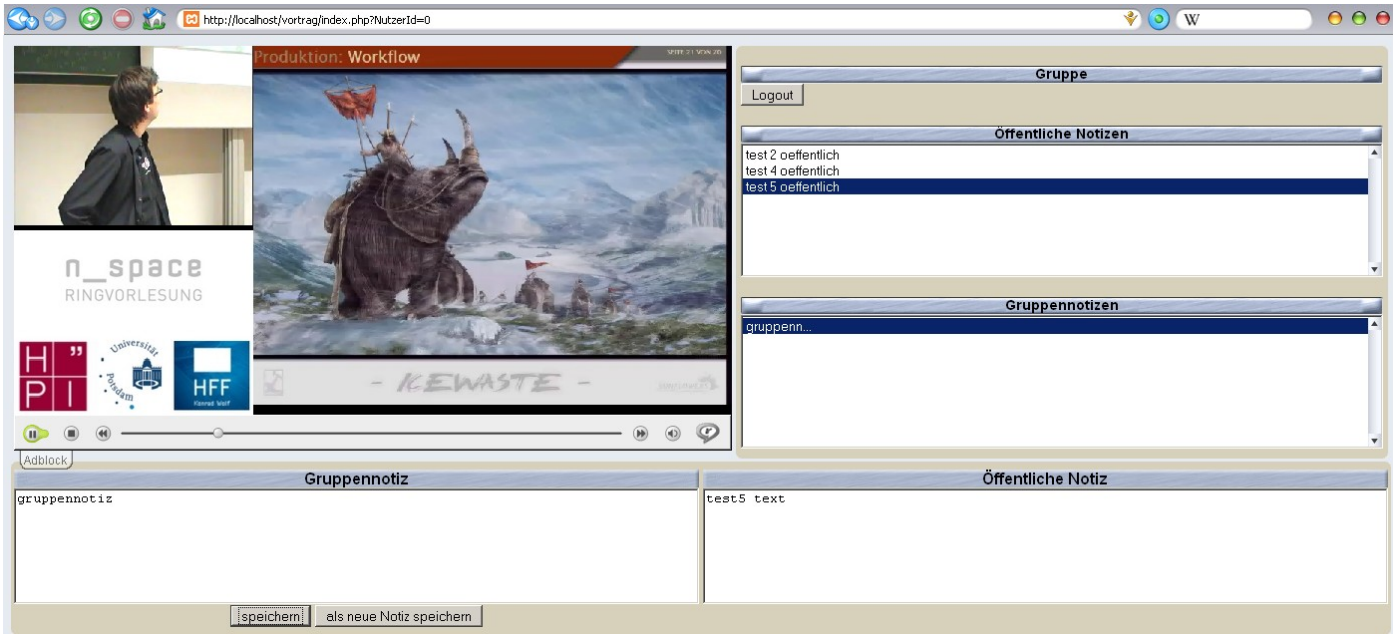

Abbildung 14: Modus: Gruppennotiz

Eine Gruppe ist jeweils für eine ganze Vorlesungsreihe gültig. Das bedeutet, dass wenn eine Gruppe während einer Vorlesung erstellt wird, diese auch für alle zugehörigen Vorlesungen in der jeweiligen Vorlesungsreihe eingerichtet ist. So kann z. B. eine Übungsgruppe zu einer Vorlesungsreihe gut abgebildet werden.

6 Zusammenfassung / Ausblick

Bisher ist es also möglich Notizen anzulegen, einem Zeitstempel zuzuordnen und diese abhängig von der aktuellen Streamposition automatisch zu laden. Außerdem gibt es bereits den Gruppenmodus, in dem Übungsgruppen gebildet werden können. Dieser ist mit Absicht ohne große Verwaltung gehalten, da die Studenten so die Möglichkeit haben, sich selbständig in Übungsgruppen zu organisieren.

Insgesamt ermöglicht das noch keine wirklich komfortable Nutzung und so wollen wir noch weitere Funktionalität hinzufügen. Zum Einen bietet es sich an, den primären Nutzern, also den Studenten, die Möglichkeit zu geben, ihre Notizen veröffentlichen zu können. Dazu soll ein Student in der Lage sein eine Notiz als „freigegeben" zu markieren.
Allerdings – um sicher zu gehen, dass auch nur nützliche Notizen veröffentlicht werden – wird eine so markierte Notiz eines Studenten noch von einem Redakteur kontrolliert. Befindet dieser sie als in Ordnung, gibt er sie endgültig frei. Der Student, der sie verfasst hat, soll dann allerdings auch nicht mehr in der Lage sein diese freigegebene Notiz zu ändern.

Des weiteren möchten wir den Gruppenmodus verfeinern. Es soll möglich sein, dass eine Gruppe einen Verantwortlichen erhält. Intuitiv ist dies derjenige, der die Gruppe anlegt, sich also zuerst mit dem Gruppennamen einlogt. Er soll die Möglichkeit bekommen Nutzer von der Gruppe zu entfernen oder auch Nutzer einzuladen, dieser Gruppe beizutreten.

Als ein letztes Gimmick werden ausgewählte Notizen als PDF exportierbar sein. Dazu wird das PDF auf dem Server erstellt und anschliessend vom Client heruntergeladen.