Ostfalia

Hochschule für angewandte Wissenschaften

Fakultät Informatik

Institut für Medieninformatik

Implementierung einer Wunschliste für ein weltweit verteiltes Assessment System

Bachelorarbeit
von

Juliane Wenzel, 20766298

Erstprüfer:	Prof. Dr. Peter Riegler, Ostfalia Hochschule
Zweitprüfer:	Prof. Dr. Gerd Kortemeyer, Michigan State University
Ort:	Wolfenbüttel
Datum:	18.08.2010

Eidesstattliche Erklärung zur Bachelorarbeit

Hiermit versichere ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen verwendet habe.

Wolfenbüttel, den 18.08.2010

_____

(Unterschrift)

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Die stetig anwachsenden Informationsmengen erfordern immer neue Ordnungsstrukturen, die Nutzern dabei helfen, den Überblick zu behalten. Im Internetbereich, in dem die Informationszunahme besonders ausgeprägt ist (vgl. [EMC]), gibt es dafür verschiedene Konzepte wie beispielsweise das des „Lesezeichens“. Diese Arbeit beschäftigt sich mit der Erstellung einer Ordnungsstruktur für den Informationsgehalt des Online-Lernsystems LON-CAPA [LON-CAPA], welches in Form von Ressourcen (Online-Aufgaben, Bilder, HTML-Seiten u.a.) in einem weltweit verteilten Netzwerk vorliegt. Ausgehend von einer bereits bestehenden Funktionalität wird das Konzept der „Wunschliste“ entwickelt. Der schriftliche Teil der Arbeit beschäftigt sich mit der Spezifikation und Beschreibung des praktischen Teils, der die Ausarbeitung im Rahmen des LON-CAPA-Systems beinhaltet.

1.1. Motivation

Die momentan in LON-CAPA implementierte Lesezeichen-Sammlung wird wenig verwendet, da sie unübersichtlich und fehlerhaft ist. Zudem ist das momentane Design der Sammlung nicht stimmig bezogen auf den Rest des Systems. Von Nutzern des Systems wurden diese Umstände bereits vor einigen Jahren über die Bugtracker-Webapplikation Bugzilla angemerkt [Bugzilla]. Die eigentliche Intention, die hinter der Sammlung steht, ist jedoch eine sehr nützliche Funktion, die den Nutzer beim Verwenden des Systems hilfreich unterstützen kann. Aus diesem Grund ist es notwendig, die benötigten Funktionalitäten zu spezifizieren und so umzusetzen, dass der Nutzer diese leicht bedienen kann und somit die ihm angebotene Unterstützung gerne annimmt.

1.2. Zielsetzung

Ziel dieser Arbeit ist es, eine Funktionalität zu schaffen, die dem Nutzer eine nützliche Unterstützung im flexiblen Zugriff auf Ressourcen bietet. Ein fehlerfreies Verhalten und eine intuitive Bedienung sind dabei die wichtigsten Aspekte. Um dies realisieren zu können, müssen die Funktionen, die dem Nutzer zur Verfügung gestellt werden, klar definiert und differenziert werden. Bei der Umsetzung ist es anschließend erforderlich, die Nutzer-Oberfläche so zu gestalten, dass sie dem Design des restlichen Systems entspricht. Durch eine konsequente und einheitliche Gestaltung der System-Oberfläche wird die Bedienbarkeit erleichtert.

1.3. Gliederung der Arbeit

Die Arbeit umfasst insgesamt sieben Kapitel. Im folgenden Kapitel 2 wird die Funktionalität von Lesezeichen erklärt. Nach einer kurzen Einführung in das LON-CAPA-System wird dann auf die Verwendung von Lesezeichen in LON-CAPA Bezug genommen. Kapitel 3 beschäftigt sich mit der Spezifikation für die Neugestaltung der Lesezeichen-Sammlung und bildet die Grundlage der Programmierarbeit. Diese werden in Kapitel 4 mit Überlegungen zur Speicherstruktur und Schnittstellen weitergeführt. In Kapitel 5 werden anschließend Entwürfe für das neue Oberflächendesign der Sammlung vorgestellt. Kapitel 6 behandelt die Vorgehensweise bei der Implementierung. Kapitel 7 rundet die Arbeit mit einer abschließenden Zusammenfassung ab.

2. Überblick

In Zeiten des immer größer werdenden Datenwachstums (vgl. [EMC]) ist es notwendig, dass die Nutzer Verweise auf für sie wesentliche Daten strukturiert abspeichern können, um einen schnellen Zugriff darauf zu habe. Dazu wird eine Funktionalität verwendet, die häufig mit „Lesezeichen setzen“ oder „bookmarken“ bezeichnet wird. Was Lesezeichen sind und wie sie im Rahmen des LON-CAPA-Systems eingesetzt werden können, wird in diesem Kapitel näher erläutert.

2.1. Lesezeichen

Lesezeichen finden besonders dort Verwendung, wo mit einer großen Menge an Daten gearbeitet wird. Dies zeigt schon die ursprüngliche, nicht-digitale Form des Lesezeichens. Meist aus Pappe dienen sie der Markierung einer Stelle im Buch, an die der Leser später zurück kehren möchte. Auch kann der Leser mehrere Stellen in einem Buch markieren und die zugehörigen Lesezeichen beschriften, um die verschiedenen Stellen unterscheiden zu können. Bei passender Beschreibung lässt sich so auch nach Jahren eine bestimmte Stelle in einem Buch wieder finden.
Das Internet stellt unter anderem eine Sammlung von Informationen dar und ist somit, im Sinne der Informationsspeicherung, ähnlich zu einem Buch. Die Menge der vorhandenen Informationen im Internet ist jedoch beachtlich größer als die eines Buches. Es ist also notwendig, dass die Nutzer eine Möglichkeit haben, Verweise auf Internetseiten, die sie später eventuell noch einmal besuchen möchten, in einer Sammlung zu hinterlegen. Jeder gängige Internet-Browser bietet diese Funktionalität. Passenderweise werden diese Verweise auch als Lesezeichen bezeichnet. Ein Lesezeichen besteht dabei aus einem Titel und der URL zur jeweiligen Internetseite. Häufig ist es möglich, zusätzlich eine Notiz hinzufügen, in der beispielsweise der Inhalt der Seite kurz zusammengefasst ist. Zudem wird eine Ordnerfunktion angeboten, sodass die Lesezeichen kategorisiert werden können. Dadurch wird eine strukturierte Speicherung von Verweisen auf für den Nutzer relevante Internetseiten und damit verbundene Informationen möglich.

2.2. LON-CAPA

Das Open-Source-System LON-CAPA („Learning*Online* Network with Computer-Assisted Personalized Approach“) entstand 1999 durch den Zusammenschluss der beiden Projekte „CAPA“ und „LectureOnline“ und kam 2001 zum erste Mal an der Michigan State University zum Einsatz. Mit der Zeit wurde LON-CAPA auch an anderen Institutionen eingesetzt, sodass sich ein weltweites Netzwerk entwickelte. Dieses Netzwerk besteht der-

zeit aus über 100 Servern, die von Institutionen wie Schulen und Hochschulen betrieben werden, ein Großteil davon in den USA.
LON-CAPA besteht aus den Komponenten Kurs-Management, Learning Content Management und automatische Aufgabenbewertung. Das System erfüllt somit mehrere Aufgaben: Als Kurs-Management-System bietet es die üblichen Funktionen wie beispielsweise Benutzerverwaltung, Gruppen- und Sektionsmanagement, Diskussion- und Chatfunktion und Kursparametrisierung. Dadurch ist eine genaue und individuelle Kursgestaltung möglich.
Der Aspekt des Learning Content Management erhält besonders durch die Vernetzung der am System beteiligten Server eine große Bedeutung. Alle Nutzer greifen auf eine gemeinsame Sammlung von Ressourcen, dem so genannten Ressourcenpool, zu. Diesen können sie durch eigene Zuarbeit erweitern, wodurch nach und nach über 400 000 Ressourcen zusammengekommen sind, die in jedem der netzwerkweiten Kurse verwendet werden können. Kurse gehören dabei immer einer bestimmten Domäne an, die üblicherweise pro Server angelegt werden. Je nach Kurseinstellungen kann ein Kurs aber für alle Benutzer des LON-CAPA-Netzwerks verfügbar sein.
Einen großen Bereich des Ressourcenpools stellen mit ca. 45% elektronische Aufgaben dar. LON-CAPA verfügt über ein Bewertungssystem, welches Antworten auf diese Aufgaben automatische auf Richtigkeit überprüft. Dadurch eröffnet sich die Möglichkeit eines schnelles Feedbacks über den Lernstand des Lernenden, der die Aufgabe beantwortet hat, sowohl für ihn selbst als auch für den Lehrenden, der die Aufgabe gestellt hat [LON-CAPA].

2.3. Lesezeichen in LON-CAPA

Wie bereits im Abschnitt 2.1 „Lesezeichen“ erwähnt wird durch Lesezeichen eine Methode geschaffen, Verweise auf Daten zu speichern. In LON-CAPA sind diese Daten die Ressourcen des Ressourcenpools. Dieser ist hierarchisch nach Domäne, Benutzer und benutzereigenen Verzeichnissen aufgebaut. Beim Einrichten von Kursen können die dort hinterlegten Ressourcen ausgewählt und verwendet werden. Es ist dabei nicht unüblich, dass ein Nutzer mehrere Kurse besitzt und manche Ressourcen in allen diesen Kursen benutzen möchte. Dies ist beispielsweise bei Aufgaben, die sich mit grundlegenden mathematischen Konzepten beschäftigen, denkbar, da diese in den verschiedensten Fachrichtungen benötigt werden. Um dem Nutzer die mehrfache Suche nach diesen Ressourcen im Ressourcenpool zu ersparen, wird eine Lesezeichen-Sammlung angeboten. Diese ermöglicht es, Pfade auf Ressourcen in einer Liste zu hinterlegen. Auf diese Liste kann aus jedem Kurs zugegriffen werden und die dort als Link hinterlegten Ressourcen in den Kurs importiert werden, sodass ein schneller Zugriff auf eine vorselektierte Untermenge des LON-CAPA Ressourcenpools möglich wird.
Neben der erleichterten Wiederverwendung gleicher Ressourcen in mehreren Kursen ist noch ein zweiter Anwendungsfall denkbar: Beim Einrichten eines Kurses könnte der Nutzer auf Ressourcen stoßen, die zwar nicht für den Kurs, in dem er sich aktuell befindet, relevant sind, jedoch sehr wohl für einen seiner anderen Kurse. In diesem Fall kann er sich

einen Link auf diese Ressourcen in der Lesezeichen-Sammlung setzen, um diese später in den entsprechenden Kurs einzubinden.

3. Spezifikation

In diesem Kapitel werden die Grundlagen für die Neu-Programmierung gelegt. Dazu ist es wichtig, möglichst genau die Funktionalität der Lesezeichen-Sammlung zu spezifizieren und abzugrenzen. Nach einer Analyse der bisherigen Funktionalität wird darauf aufbauend die Spezifikation der neuen Lesezeichen-Sammlung entwickelt.

3.1. Analyse der alten Funktionalität

Die Lesezeichen-Sammlung wird über einen Menüpunkt im Hauptmenü aufgerufen. Abbildung 3.1 zeigt das Pop-Up-Fenster, welches die Sammlung visualisiert.

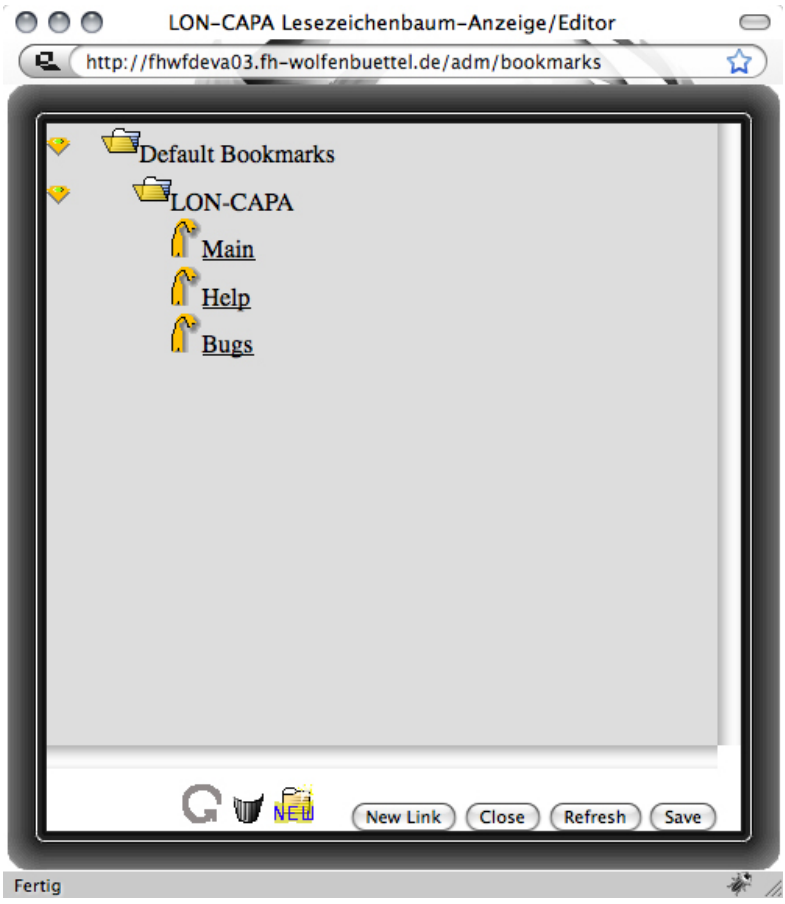

Abbildung 3.1.: Lesezeichen-Sammlung in LON-CAPA (altes Design)

Dieses Fenster gliedert sich in zwei Bereiche. In dem oberen Teil ist die bereits angesprochene Ordnerstruktur zu erkennen, mit Hilfe derer eine Kategorisierung der Lesezeichen vorgenommen werden kann. Über die Pfeile vor den Ordnersymbolen lassen sich die einzelnen Ordner auf- und zuklappen. Im unteren Bereich des Fensters befinden sich die Funktionen zum Editieren der Sammlung. Diese umfassen das Anlegen neuer Ordner (Ordner-Icon), das Anlegen neuer Lesezeichen mit Titel und Pfad („New Link“) sowie das Löschen bereits existierender Lesezeichen (Mülleimer-Icon). Änderungen werden erst nach einer Bestätigung über „Save“ übernommen. „Refresh“ aktualisiert die Ansicht, über „Close“ lässt sich das Fenster schließen.
Der Titel eines Eintrages dient als Link zu der hinterlegten Ressource. Zum Betrachten muss dieser Link betätigt werden, im Hauptfenster wird dann der Inhalt der Ressource angezeigt.
Lesezeichen lassen sich nicht nur bei der Betrachtung der Sammlung durch „New Link“ hinzufügen, sondern auch beim Betrachten einer Ressource, die in einem Kurs eingebunden ist. Dazu steht dem Nutzer ein Lesezeichen-Icon zur Verfügung. Im Anlege-Dialog sind die Felder für Titel und Pfad dabei bereits ausgefüllt (Titel ist der in den Meta-Daten der Ressource hinterlegte Ressourcen-Titel), können jedoch vom Nutzer geändert werden.
In der Sammlung referenzierte Ressourcen lassen sich in einen Kurs importieren. Dazu wird dem Nutzer ein Dialog angeboten, in dem er die zu importierenden Ressourcen auswählen kann. Abbildung 3.2 zeigt diesen Dialog:

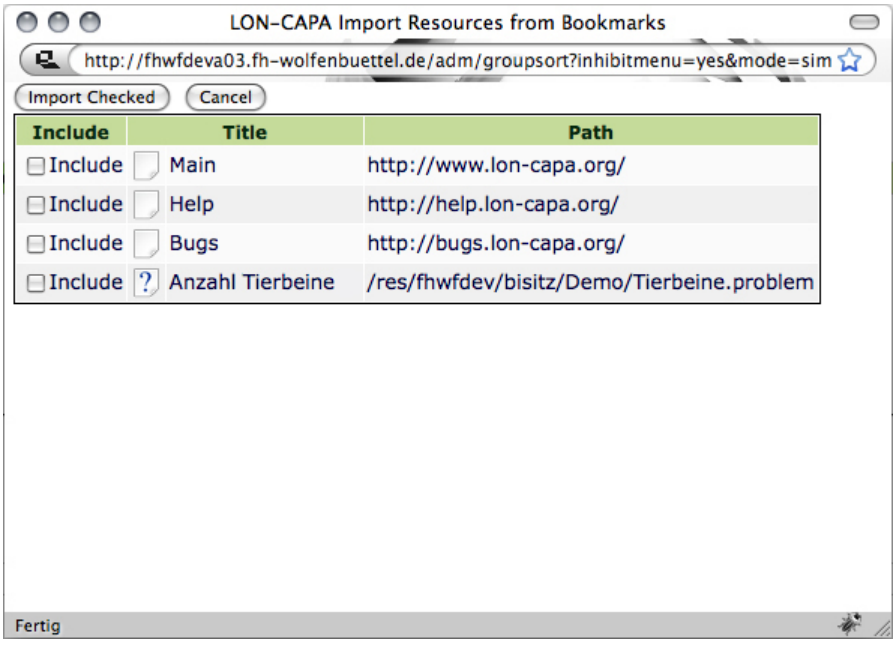

Abbildung 3.2.: Ressourcen aus der Lesezeichen-Sammlung in einen Kurs importieren

Nutzer, die noch keine Lesezeichen angelegt haben, finden in ihrer Sammlung so genannte

„Default-Lesezeichen“ (wie in Abbildung 3.1 zu sehen) vor. Diese werden auch automatisch generiert, sollte ein Nutzer alle seine Lesezeichen gelöscht haben. Die Sammlung wird somit nie komplett leer sein.

3.1.1. Fehlerhafte und umständliche Funktionen

Einige der Funktionen, die die Lesezeichen-Sammlung bietet, sind momentan fehlerhaft oder für den Nutzer umständlich zu bedienen. Im Folgenden werden diese Schwachstellen aufgezeigt.
Zum Betrachten seiner Lesezeichen-Sammlung wählt der Nutzer im Hauptmenü den entsprechenden Menüpunkt. Das sich dann öffnende Fenster (vgl. Abbildung 3.1) ist allerdings nicht groß genug, um alle enthaltenen Funktionen direkt erkennen zu können. Um Zugriff darauf zu erhalten, muss der Nutzer das Fenster vergrößern. Auch ist die Bedienung der Funktionen nicht intuitiv. Nach dem Hinzufügen eines neuen Lesezeichen oder dem Löschen eines bereits vorhandenen, ist ein expliziter Klick auf „Save“ notwendig, damit die Änderungen übernommen werden. Hinzu kommt, dass die Vorgehensweise beim Löschvorgang an sich nicht direkt erkennbar ist. Zum Löschen muss der Nutzer erst auf das Icon vor dem Lesezeichen klicken und dann auf das Mülleimer-Icon. Fährt der Nutzer jedoch nach dem Klick auf ein Lesezeichensymbol mit der Maus über die anderen (Lesezeichen- oder Ordner-) Symbole, so führt dies zu einem eigenartigen Effekt: jedes Symbol wechselt für den Moment, in dem der Mauspfeil auf ihm verharrt in das jeweils andere Symbol. Ordnersymbole werden also zu Lesezeichensymbolen und andersrum. Ein Doppelklick auf eine Ordnersymbol lässt diesen sogar mit samt dem Inhalt aus der Liste verschwinden. Speichert der Nutzer in diesem Zustand, so sind die Lesezeichen tatsächlich gelöscht.
Wird ein neues Lesezeichen über „New Link“ hinzugefügt, so wird dieses immer ganz unten in der Liste angezeigt und befindet sich außerhalb jedes Ordners. Das Verschieben von Lesezeichen ist umständlich und für den Nutzer ist auf Anhieb nicht zu erkennen, wie dieses überhaupt funktioniert. Denn um ein Lesezeichen zu verschieben muss der Nutzer zuerst auf das Symbol vor dem Lesezeichen klicken und dann an die Stelle, an die es verschoben werden soll, also beispielsweise auf ein Ordnersymbol. Auch ganze Ordner können so samt Inhalt verschoben werden. Jedoch ist das bereits beschriebene Verhalten beim Klicken auf Symbole (Symbolwechsel, Löschen des Ordners) für den Nutzer verwirrend. Ein Klick zu viel führt dazu, dass Lesezeichen und Ordner gelöscht werden. Insgesamt ist das Verhalten beim Verschieben nicht konsequent und somit für den Nutzer auch nicht nachvollziehbar.
Beim Anlegen eines Lesezeichens findet keine Prüfung bezüglich des Pfades statt. Dies ist gerade deswegen problematisch, weil es auch möglich ist, Lesezeichen für externe Seiten zu hinterlegen. Pfade, die zu externen Seiten führen, müssen jedoch absolut, also mit entsprechendem Netzwerkprotokoll angegeben werden. Eine Eingabe ohne Protokoll führt dazu, dass der angegeben Pfad relativ auf das LON-CAPA-Netzwerk bezogen wird. Es wird also in dem Ressourcenpool nach dem Pfad gesucht. Da dieser aber nicht gefunden wird, erhält der Nutzer beim Betrachten des angelegten Links eine Fehlermeldung darüber, dass die Ressource nicht verfügbar ist. Dieses Verhalten ist zwar nach-

vollziehbar, allerdings nicht vorhersehbar und könnte deswegen den Nutzer beim ersten Vorkommen verwirren.
Momentan ist es nicht möglich, Lesezeichen direkt aus der Ressourcen-Suche heraus zu setzen. Der Pfad zu einer Ressource muss per Hand kopiert, ein neues Lesezeichen in der Sammlung angelegt und dabei der kopierte Pfad eingefügt werden. Dabei muss auf die Korrektheit geachtet werden, da, wie bereits erwähnt, keine Prüfung der angegebenen Pfade vorgenommen wird. Rechtschreibfehler führen damit zu ungültigen Lesezeichen, die beim Importieren in einen Kurs Fehler erzeugen. Da eine nachträgliche Änderung von Titel und Pfad eines Lesezeichens nicht möglich ist, ist im Falle eines Schreibfehlers ein Löschen des bereits angelegten Lesezeichens und das Anlegen eines Neuen notwendig.
Beim Importieren von Lesezeichen wird aktuell eine Liste der angelegten Lesezeichen mit Titel und Pfad zur Ressource angezeigt. Die Ordnerstruktur wird dabei nicht berücksichtigt. Dadurch gibt es momentan auch keine Möglichkeit, alle Lesezeichen aus einem bestimmten Ordner auf einmal zu importieren. Der Nutzer muss einzeln per Hand die Lesezeichen auswählen, die er importieren möchte.

3.2. Spezifikation der zukünftigen Funktionalität

Bei der Analyse der bisherigen Funktionen wird deutlich, dass grundsätzlich zwischen drei Kategorien von Links, die in der Sammlung gespeichert werden können, unterschieden werden muss:

1. Links, die Pfaden auf Ressourcen im Ressourcenpool entsprechen,
2. Links auf externe Webseiten,
3. Links auf Kursinhalte

Beim Speichern und Betrachten von Lesezeichen ergeben sich sowohl bei Kategorie eins als auch Kategorie zwei keine Probleme. Sollte eine Webseite nicht mehr existieren, so erhält der Nutzer lediglich eine Meldung darüber und kann dann wie gewohnt fortfahren. Ressourcen aus dem LON-CAPA-Ressourcenpool können, wenn sie einmal veröffentlicht sind, weder verschoben noch gelöscht werden. Sie können somit auch immer unter dem einmal angelegten Link betrachtet werden. Vor allem aber können Ressourcen aus dem Ressourcenpool aus jedem Kurs heraus betrachtet werden, solange die verknüpfte Kurs-Rolle den Zugriff auf den Ressourcenpool erlaubt. Anders verhält sich dies bei Kursinhalten, also beispielsweise bei einer von der Festplatte in den Kurs geladene PDF-Datei oder einer in dem Kurs erzeugten „Einfachen Aufgabe“. Links, die auf diese Kursinhalte zeigen, besitzen ihre Gültigkeit nur innerhalb dieses Kurses. Beim Betrachten eines solchen Links müsste also der Kurs gewechselt werden, sofern es sich um einen Link auf den Inhalt eines anderen als den derzeit ausgewählten Kurs handelt. Würde der Kurs nicht gewechselt, so erhielte der Nutzer eine Meldung darüber, dass er keine Zugriffrechte für die angeforderte Seite besitzt.

Auch beim Importieren von Lesezeichen in einen Kurs ist eine Unterscheidung der genannten Kategorien notwendig. Sowohl Lesezeichen, die auf Ressourcen verweisen, als auch die, die auf externe Webseiten verlinken, können bedenkenlos importiert werden. Da ein Ressourcen-Pfad niemals seine Gültigkeit verliert, führt das Importieren eines solchen Lesezeichens immer dazu, dass eine gültige LON-CAPA-Ressource in den Kurs eingebunden wird. Auch das Einbetten von externen Seiten in einen Kurs führt dazu, dass ein gültiges Dokument eingebunden wird. Sollte die angegeben Seite nicht mehr existieren oder eine Authentifizierung erwarten, so wird dem Nutzer genau dieses angezeigt, da die Seite in einem Frame komplett eingebettet wird. Das Importieren von Lesezeichen, die auf Kursinhalte zeigen, kann jedoch zu einem Fehler führen. Dies ist aus demselben Grund der Fall, aus dem auch das Betrachten von Links auf Kursinhalten unter den genannten Umständen fehlschlägt. Das Anlegen von Links auf Kursinhalte sollte also nicht gestattet sein, um diesen Konflikt zu vermeiden.
Die Funktionalität, die dem Nutzer geboten wird, ähnelt der, die üblicherweise in Browsern als „Lesezeichen-Funktionalität“ bezeichnet wird. Der entscheidende Unterschied ist jedoch, dass die Liste nur innerhalb des LON-CAPA-Netzwerkes verfügbar ist und die vermerkten Einträge nicht nur betrachtet, sondern auch in einen Kurs importiert werden können. Um dieses genügend zu differenzieren und dem Nutzer auch rein sprachlich die Unterscheidung deutlich zu machen, wird die zukünftige Funktionalität in LON-CAPA nicht mehr mit „Lesezeichen-Sammlung“ betitelt, sondern mit „Wunschliste“. Auf dieser kann ein Nutzer Links auf LON-CAPA-Ressourcen oder externe Webseiten hinterlegen, die er später noch einmal zu betrachten oder in einen Kurs zu importieren wünscht.[1]
Die Wunschliste wird nur für Benutzer-Rollen mit Zugriff auf den Ressourcenpool verfügbar sein. Funktional wird eine Möglichkeit geschaffen, Links auf Ressourcen abzuspeichern, um diese weiter zu verarbeiten. Dabei verweist der Link immer auf eine Ressource im Ressourcenpool oder auf eine externe Webseite. Die Links in der Wunschliste sind somit unabhängig von Kursen. Der Hauptnutzen wird das Speichern zum späteren Importieren der hinterlegten Ressourcen in einen Kurs sein. Die Wunschliste erleichtert dem Nutzer dabei das Wiederverwenden von bereits genutzten Ressourcen. Im Folgenden wird dies ausführlich erläutert.
Die Wunschliste beinhaltet Links mit den obligatorischen Informationen „Titel“ und „Pfad“ sowie eine optionale Notiz. Die Notiz ermöglicht dem Nutzer, die verknüpfte Ressource zu beschreiben. Strukturiert wird die Liste über eine Ordnerstruktur, wobei Ordner ebenfalls mit Notizen versehen werden können. Somit sollte zwischen drei Funktionsbereichen unterschieden werden:

Funktionen, die auf die gesamte Wunschliste wirken:

- Einen neuen Ordner erstellen.
- Einen neuen Link erstellen

[1] Der Begriff „Wunschliste“ beschreibt die Funktionalität derzeit am besten und wird auch für die Nutzeroberfläche verwendet. Er ist dort jedoch leicht austauschbar, falls dies bei späteren Weiterentwicklungen notwendig oder gewünscht ist.

  - über manuelles Eingeben von Titel und Pfad.
  - über das Betrachten einer Ressource.

Funktionen, die auf Ordner ausgeführt werden:

- Ordner-Details (Name, Notiz) betrachten.
- Ordner-Details verändern.
- Ordner-Reihenfolge umsortieren.
- Einen oder mehrere Ordner verschieben.
- Einen oder mehrere Ordner löschen.

Funktionen, die auf einzelne oder mehrere Links ausgeführt werden:

- Einen Link betrachten.
- Link-Details (Titel, Pfad, Notiz) betrachten.
- Link-Details ändern.
- Link-Reihenfolge umsortieren.
- Einen oder mehrere Links verschieben.
- Einen oder mehrere Links löschen.
- Einen oder mehrere Links in einen Kurs importieren.

Ein Link in der Wunschliste kann auf zwei Arten erzeugt werden: Entweder durch manuelle Nutzereingabe, welche beispielsweise bei Links auf externen Seiten notwendig ist, oder, für Links auf LON-CAPA-Ressourcen, beim Betrachten einer solchen. Bei jeder „Vorschau“ einer Ressource, sei es aus der Ressourcen-Suche heraus oder beim Betrachten veröffentlichter Ressourcen, wird dem Nutzer dazu ein Icon angeboten, über welches direkt ein Link auf die betrachtete Ressource in die Wunschliste gesetzt werden kann. Das Feld für den Titel ist dabei bereits mit dem Titel, der in den Meta-Daten der Ressource hinterlegt ist, ausgefüllt. Der Nutzer kann diesen vor dem Speichern des Links ändern und eine Notiz hinzufügen. Der Pfad zur Ressource wird automatisch gesetzt. Alle Informationen sind im Nachhinein editierbar.
Der Nutzer kann beim Anlegen eines Links direkt entscheiden, in welchen Ordner dieser hinterlegt werden soll. Auch Ordner können beim Erstellen als Unterordner eines bereits existierenden Ordners erzeugt werden.
Ein gerade erstellter Link oder Ordner wird immer als letzter Eintrag in dem ausgewählten Ordner angefügt. Der Nutzer kann die Reihenfolge der Einträge jedoch über eine Sortier-Funktion im Nachhinein ändern. Sortiert werden können immer nur die Einträge innerhalb eines Ordners und die Einträge auf der obersten Ebene der Liste. Zusätzlich ermöglicht eine Verschiebe-Funktion das Verschieben von Einträgen in einen anderen

Ordner, welcher sich auch auf einer anderen Ebene befinden kann. Beim Verschieben von Ordnern wird automatisch der Ordnerinhalt mit verschoben. Ein Ordner kann somit nicht in einen seiner Unterordner verschoben werden.
Bei der manuellen Erstellung eines Links ist eine Prüfung des eingegebenen Pfades notwendig. Sollte der Pfad nicht zu einer LON-CAPA-Ressource oder einer externen Webseite gehören, so wird der Nutzer darauf hingewiesen. Der eingegebene Pfad kann dann noch einmal überprüft und geändert werden.
Beim Importieren von Links in einen Kurs sollte der Auswahl-Bildschirm, auf dem die zu importierenden Links ausgewählt werden, dieselbe Struktur aufzeigen, wie die Wunschliste. Dies erleichtert dem Nutzer die Orientierung innerhalb seiner angelegten Links. Auch wird es möglich sein, alle Links eines Ordners zu importieren, ohne dass diese einzeln markiert werden müssen. Wählt der Nutzer einen Ordner aus, so werden automatisch alle enthaltenen Links für den Import ausgewählt. Außerdem sollen über einen „Alle auswählen“-Button alle Links auf einmal markiert werden können. Um alle Markierungen aufzuheben, wird ein „Alle aufheben“-Button angeboten. Die Links werden mit dem in der Wunschliste hinterlegten Titel in den Kurs importiert.

3.3. Abgrenzung - Was nicht angedacht ist

In diesem Abschnitt sollen denkbare Funktionalitäten der Wunschliste erläutert werden, die zusätzliche angeboten werden könnten, jedoch nicht essentiell für die Nutzung der Wunschliste sind und somit nicht im Zuge der Neugestaltung implementiert werden. Grundsätzlich wird bei der Implementierung auf Erweiterbarkeit geachtet, so dass diese Funktionen bei Bedarf später realisiert werden können.
Eine Suche innerhalb der Wunschliste ist für die Neugestaltung nicht vorgesehen. Bei längeren Wunschlisten mit sehr vielen Links wäre dies aber eine durchaus nützliche Funktion. Hilfreich könnten in diesem Fall auch so genannte *Tags* sein, bei denen ein Link mit einem Schlagwort versehen wird. Dies macht eine noch genauere Kategorisierung, als die durch die Ordnerstruktur gegebene, möglich.
Weiterhin ist keine Prüfung eines neu zu erstellenden Links gegen die bereits existierenden geplant. Der Nutzer kann für einen Ressourcen-Pfad also mehrere Links anlegen. Das System fängt dies nicht ab, da es unter Umständen sogar ein von dem Nutzer gewolltes Verhalten ist. Der Nutzer trägt in diesem Fall selbst Sorge für die Ordnung seiner Sammlung.
Zusätzlich zu den bisher erwähnten Möglichkeiten, einen Link in der Wunschliste zu setzen, ist noch eine weitere denkbar. Kurse in LON-CAPA enthalten oftmals zum großen Teil Ressourcen aus dem Ressourcenpool. Statt diese beim Betrachten einzeln auf die Wunschliste zu setzen, könnte auch eine Export-Funktion angeboten werden. Diese filtert den Kurs-Inhalt und sortiert alle Inhalte aus, die keine LON-CAPA-Ressourcen oder Links zu externen Webseiten sind und behält dabei die Ordnerstruktur des Kurses bei. Der Nutzer hat dann die Möglichkeit, einzelne Einträge oder ganze Ordner aus der Export-Liste auf die Wunschliste zu übertragen. Dadurch können einmal zusam-

mengestellte Ordner aus einem Kurs über Export und Import in einen anderen Kurs übernommen werden. In LON-CAPA werden bereits zwei Möglichkeiten angeboten, die eine ähnliche Funktion wie die gerade beschriebene bieten. Dazu zählt zum einen der Export und Import von so genannten IMS-Paketen. Dieses Datenformat erlaubt es zudem, Inhalte über verschiedene Lernsysteme hinweg auszutauschen. Zum anderen existiert ein Mechanismus, der es erlaubt, die Inhalte von Kursen in den Konstruktionsbereich zu laden. Dort können diese dann bearbeitet und für die spätere Wiederverwendung veröffentlicht werden.
Wunschlisten sind personenbezogen und stehen somit nur jedem Nutzer einzeln zur Verfügung. Nutzer könnten jedoch die Möglichkeit eines Austausches von Wunschlisten beziehungsweise -einträgen untereinander wünschen. Technisch könnte diese Austauschmöglichkeit durch den Export ausgewählter Einträge in eine Datei und den Import dieser Datei in eine andere Liste realisiert werden. Dazu würde sich beispielsweise eine XML-Struktur anbieten. Ein anderer Ansatz wäre eine direkte Zugriffsmöglichkeit auf Wunschlisten anderer Nutzer. Dazu müsste ein Nutzer bestimmen können, welche anderen Nutzer des Netzwerkes für welchen Zeitraum Zugriff auf welche Einträge aus seiner Liste haben.

4. Speicherstruktur und Schnittstellen

In diesem Kapitel wird erläutert, wie die zugrundeliegenden Daten der Wunschliste system-intern strukturiert und gespeichert werden. Zudem werden die Schnittstellen der Wunschliste zu anderen LON-CAPA-Komponenten definiert.

4.1. Speicherung der Daten

Benutzerdaten werden in LON-CAPA in db-Datenbank-Dateien gespeichert. Für den Zugriff auf diese Dateien gibt es die Funktionen „put“ und „get“ beziehungsweise „dump“. Als Parameter wird der Funktion „put“ der eigentlich zu speichernde Inhalt in Form eines Hashes und ein Dateiname übergeben. Der Hash wird dann in der db-Datei mit dem angegebenen Dateinamen gespeichert. Über „dump“ kann der gesamte Inhalt einer db-Datei ausgelesen werden. Über „get“ lassen sich Teile der db-Datei auslesen, indem die Keys zu den gewünschten Einträgen übergeben werden. Weiterhin gibt es die Funktionen „keys“, um alle in der db-Datei eingetragenen Keys auszugeben, und „del“ zum Löschen von Einträgen in der db-Datei.
Die Wunschliste ist personengebunden und somit erfolgt ihre Speicherung als Benutzerdaten für jeden Nutzer separat. Bei den zu speichernden Daten handelt es sich um die angelegten Ordner (mit Attributen „Titel“ und „Notiz“) und Links (mit Attributen „Titel“, „Pfad“ und „Notiz“) sowie einer Vorschrift, mit der die Verzeichnis-Struktur der Liste aufgebaut wird. Diese Daten müssen letztendlich in die Struktur eines Hashes gebracht werden, um diesen über die Zugriffsfunktionen „put“ und „dump“ in die db-Datei zu schreiben beziehungsweise daraus zu lesen.

4.1.1. Aufbau der Speicherstruktur

Um die Struktur einer Wunschliste zu verdeutlichen ist es hilfreich, diese anhand eines Beispiels zu veranschaulichen. Abbildung 4.1 beinhaltet eine solche Beispielliste. Die Ordner-Icons zeigen dabei alle geöffneten Ordner, wodurch symbolisiert wird, dass sich keine weiteren Ordner oder Links innerhalb der Wunschliste befinden. Die Beispielliste besteht also aus 3 Ordnern und 4 Links.

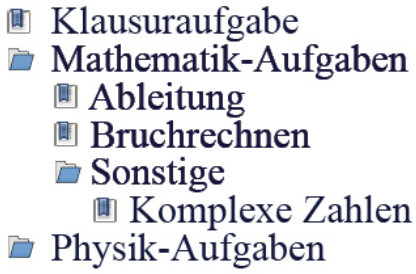

Abbildung 4.1.: Beispiel einer Wunschliste

Ausgehend von der obersten Ebene können durch das Anlegen von Ordnern und Unterordnern beliebig tiefer liegende Ebenen erzeugt werden. Es bietet sich daher an, diese Struktur in einem Baum abzubilden. Jeder Eintrag in der Liste wird dabei in einen Knoten des Baumes gespeichert. Die Blätter des Baumes stellen Links oder leere Ordner dar, während es sich bei den inneren Knoten um nicht-leere Ordner handelt. Jedoch gibt es in der Verzeichnisstruktur der Wunschliste keine eindeutige Wurzel, da sich auf der obersten Ebene mehrere Ordner und/oder Links befinden können. Vielmehr handelt es sich um mehrere Bäume, von denen manche aus nur einem Knoten bestehen. Dies ist bei Links oder leeren Ordnern, die sich auf der obersten Ebene befinden, der Fall. Um diese Teilbäume zusammenzusetzen und einen einzigen, vollständigen Baum konstruieren zu können, wird ein zusätzlicher Knoten eingeführt, der nicht direkt Bestandteil der Wunschliste ist. Dieser Knoten fungiert als Wurzel, seine Nachfolger sind die Einträge, die sich auf der obersten Verzeichnisebene der Liste befinden. Abbildung 4.2 zeigt die Umsetzung der Beispielliste in eine Baumstruktur:

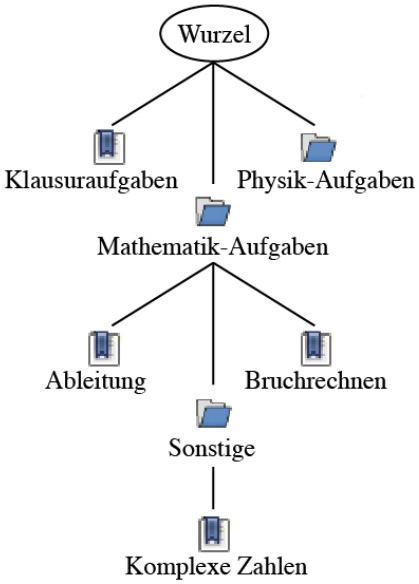

Abbildung 4.2.: Baumstruktur der Beispielliste

Jeder Knoten dieses Baumes, mit Ausnahme der Wurzel, ist ein Eintrag der Liste und muss somit folgende Daten speichern:

- Der Titel, der dem Nutzer in der Liste angezeigt werden soll.
- Einen Pfad, der nur gesetzt ist, wenn es sich bei dem Eintrag um einen Link handelt und der dann dem Pfad der Ressource im Pool oder dem Pfad zu einer externen Webseite entspricht. Ist das Attribut nicht gesetzt, so handelt es sich um einen Ordner.
- Eine Notiz, die optional für einen Eintrag angelegt werden kann.
- Zusätzlich zu den Daten, die für den Nutzer sichtbar sind, wird intern auch das Datum, an dem der Eintrag erzeugt wurde, gespeichert.

Hinzu kommen Funktionen, die auf einen Eintrag angewendet werden können. Dazu gehört in erster Linie das Erzeugen eines neuen Eintrages durch entsprechende Eingabe der benötigten Daten. Ist ein Eintrag angelegt, so kann dieser im Nachhinein editiert, das heißt, die beim Anlegen eingegeben Daten verändert werden. Einträge können darüber hinaus auch umsortiert und verschoben werden, sodass eine dynamische Strukturierung der Wunschliste ermöglicht wird. Ein nicht mehr benötigter Eintrag kann zudem wieder aus der Liste gelöscht werden.
Da die Wurzel nur ein zusätzlicher, zur Bildung des Baumes notwendiger Knoten ist, der aber keinen Eintrag in der Wunschliste bildet, besitzt er als einziger nicht die oben aufgeführten Daten. In ihm müssen nur seine Nachfolger gespeichert werden. Auch gelten die erwähnten Funktionen für ihn nicht.

Die Umsetzung der beschrieben Struktur erfolgt über einen objektorientierten Ansatz. Die Einträge der Wunschliste werden dabei als Objekte modelliert, die die eigentlichen Daten enthalten. Über eine weitere Klasse werden Objekte erzeugt, die die Baumstruktur gestalten. Die Klassen werden mit „Entry“ beziehungsweise „Tree“ bezeichnet. Die Beziehung zwischen Objekten dieser Klassen ist immer 1:1, denn ein „Tree“-Objekt, welches einen Knoten des Baumes darstellt, enthält als Attribute jeweils genau ein Objekt der Klasse „Entry“. Abbildung 4.3 visualisiert dies durch ein vereinfachtes Klassendiagramm.

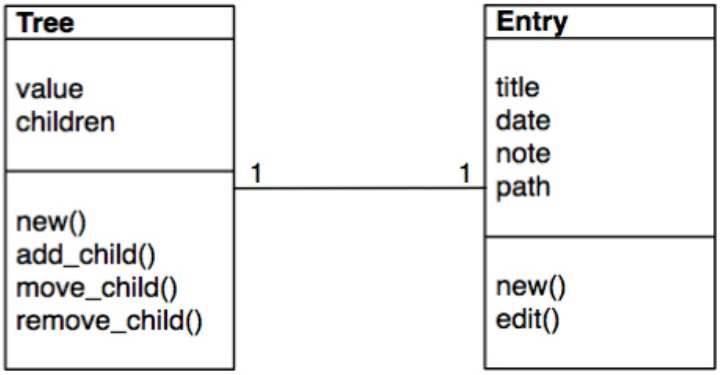

Abbildung 4.3.: Klassendiagramm

Ein „Tree“-Objekt ist ein Knoten in dem Verzeichnis-Baum. Über die Methode add_child() können einem Knoten Nachfolger („children“) hinzugefügt werden. Durch das Attribut „value“ wird der Inhalt eines Knotens definiert. Da der Wurzel-Knoten ein spezieller Knoten ist und nicht die Daten eines Wunschlisteneintrages speichern muss, erhält er als „value“ den Wert „root“. Alle anderen Knoten sind Wunschlisteneinträgen und beinhalten als Attribute somit ein „Entry“-Objekt.
Wird ein Eintrag in der Wunschliste hinzugefügt, so wird ein „Entry“-Objekt mit den eingegebenen Daten angelegt. Dieses „Entry“-Objekt wird einem neu angelegtem „Tree“-Objekt als value-Attribut übergeben. Das „Tree“-Objekt wird entsprechend der Stelle, an der der Eintrag in der Wunschliste eingetragen werden soll, einem anderen „Tree“-Objekt als Nachfolger übergeben.
Beim Verschieben von Einträgen über die Methode move_child() sind drei Schritte notwendig: Für jeden zu verschiebenden Ordner und jeden Link, der einzeln verschoben werden soll, müssen die dazugehörigen „Tree“-Objekte kopiert und über die Methode remove_child() aus dem Baum entfernt werden. Danach müssen die kopierten Objekte an der neuen Stelle im Baum über add_child() wieder hinzugefügt werden. Soll ein Eintrag gelöscht werden, so werden auch alle eventuell vorhandenen Nachfolger gelöscht,

denn das Löschen eines Ordners impliziert immer auch das Löschen des Ordnerinhaltes. Im Gegensatz zu den bisher beschriebenen Funktionen hat das Editieren eines Eintrages keine Auswirkungen auf die Baumstruktur sondern lediglich auf die Werte der Attribute des „Entry“-Objekts eines Knoten. Deshalb wird diese Methode in der Klasse „Entry“ definiert.
Durch die Trennung von Baumstruktur und Daten in zwei separate Klassen bleibt die Struktur für eventuelle spätere Änderungen erweiterbar. So ist es denkbar, dass eine Funktionalität gewünscht ist, die Links auf Seiten innerhalb eines Kurses speichert. Diese Seiten können einem temporären Zugriff unterliegen und sind darüber hinaus nicht an einen bestimmten Pfad innerhalb eines Kurses gebunden. Eine Ressource, die in einen Kurs eingebunden ist, kann nachträglich verschoben werden. Dies ist bei Ressourcen im Ressourcenpool nicht der Fall. Links auf Seiten im Kurs müssen deshalb anders gehandhabt werden als Links auf Ressourcen im Pool. Die beschriebene Speicherstruktur ermöglicht es, eine neue Klasse für solche Seiten zu definieren und daraus erzeugte Objekte an ein „Tree“-Objekt zu übergeben und in die Verzeichnisstruktur einzubetten. Über die Klasse ergäbe sich dann eine Filtermöglichkeit, um die verschiedenen Typen unterschiedlich zu handhaben.

4.1.2. Umsetzung in einen Hash

Wie in der Einleitung dieses Kapitels erläutert, ist es für die Speicherung der Struktur notwendig, diese auf einen Hash abzubilden. Zu diesem Zweck wird pro Eintrag in der Wunschliste ein Hash-Eintrag gebildet.
Damit jeder Knoten im Baum eindeutig identifizierbar ist, erhält er einen Index. Ausgehend von der Wurzel, die den Index „root“ erhält, werden dafür alle Knoten durchnummeriert. Dieser Index wird für den Key eines Hash-Eintrages verwendet. Der Value enthält zum einen alle Daten eines Wunschlisten-Eintrages (Titel, Pfad, Notiz und Anlege-Datum) und zum anderen die Indizes der Nachfolger-Knoten.
Nach dem Auslesen des Hashes aus der Datenbank muss dieser wieder in die entsprechenden Objekte umgewandelt werden. Aus jedem Eintrag des Hashes muss ein „Tree“- und das dazugehörende „Entry“-Objekt erzeugt werden. In den ersten vier Werten des Values stehen die Daten Titel, Pfad, Notiz und Anlege-Datum. In den nächsten Werten des Array befinden sich die Indizes eventueller Nachfolger des Eintrages. Der Index eines „Tree“-Objekts steht wie beschrieben in dem Key des Hash-Eintrages. Die Objekte werden wie folgt erzeugt:

Für jeden Hash-Eintrag

1. Erzeuge ein „Tree“-Objekt, bei dem der Index auf den Wert des Keys gesetzt wird. Wenn der Index „root“ entspricht, speichere den Knoten separat ab und überspringe die nächsten Punkte.

2. Erzeuge aus den ersten vier Werten des Arrays ein „Entry“-Objekt.

3. Übergebe dieses Objekt dem „Tree“-Objekt als value-Attribut.

4. Speichere das „Tree“-Objekt in einem Array.

Dieses Array enthält dann alle Knoten, die für den Aufbau des Baumes benötigt werden. Um die Verknüpfungen zwischen den Knoten zu bilden, sind folgende Schritte notwendig:

1. Beginne mit dem Wurzel-Knoten.

2. Bestimme über die restlichen Werte des Value-Arrays die Indizes der Nachfolger-Knoten.

3. Für jeden Index, suche in dem Knoten-Array nach dem entsprechenden „Tree“-Objekt des Index und füge diese über add_child() der Wurzel hinzu.

4. Verfahre jetzt für jeden Nachfolger-Knoten wie mit der Wurzel und ersetzte dabei den Wurzel-Knoten in Punkt 3 durch den aktuellen Knoten.

Über diesen Algorithmus kann aus dem Hash wieder ein vollständiger Baum konstruiert werden.

4.2. Schnittstellen zu anderen Komponenten von LON-CAPA

Die Wunschliste ist eine Software-Komponente des LON-CAPA-Systems. Die Daten, die diese Komponente ausmachen, können auf zwei verschiedene Weisen erzeugt werden:

1. Anlegen eines Links durch manuelle Eingabe der benötigten Daten.

2. Setzen eines Links für eine bestimmte Ressource über Betrachten dieser Ressource.

Im ersten Fall ist keine Schnittstelle zu anderen LON-CAPA-Komponenten notwendig. Es wird lediglich ein Eingabefeld als Benutzerschnittstelle benötigt, in dem der Nutzer die Daten eingeben kann. Beim zweiten Fall verhält sich dies anders. Betrachtet der Nutzer eine Ressource, so kann er für diese direkt einen Link setzen. Dabei sollen Titel und Pfad der Ressource automatisch gesetzt werden. Es ist also notwendig, dass zum einen dem Nutzer durch einen entsprechenden Link oder ein Icon die Möglichkeit geboten wird, einen Eintrag zu erstellen. Zum anderen müssen dafür die benötigten Informationen über die Ressource bereitgestellt und verarbeitet werden.
Eine weitere Schnittstelle wird beim Importieren von Links in einen Kurs notwendig. An dieser Stelle werden Daten aus der Wunschliste an ein anderes Modul übergeben. Dazu müssen die Daten entsprechend aufbereitet und in der erwarteten Form weitergegeben werden, so dass sie erfolgreich in den Kurs importiert werden können.
Die beschriebene Speicherstruktur erleichtert den Zugriff auf die hinterlegten Daten. Sobald die Baumstruktur erzeugt wurde, können über das Traversieren des Baumes ausgehend von der Wurzel sämtliche Daten ausgelesen werden. Auch das Hinzufügen von neuen Daten ist unkompliziert. Es werden nur die entsprechenden Attribute benötigt,

um einen Knoten zu erzeugen und die Information, an welcher Stelle im Baum dieser neue Knoten eingefügt werden soll. Dabei spielt es zudem keine Rolle, ob der neue Knoten ein Blatt ist oder die Wurzel eines Baumes, denn auch Unterbäume können in den existierenden Baum eingehängt werden. Dies bedeutet für den Nutzer, dass es möglich ist, einen Ordner mit Inhalt in seine Wunschliste einzufügen. Dies ist momentan nicht vorgesehen, ist jedoch bei einer gewünschten Umsetzung, beispielsweise durch die in Abschnitt 3.3 dargestellte Export-Funktion, auf Grund der beschriebenen Fakten leicht zu realisieren.

5. Oberflächendesign

Das Design einer Benutzeroberfläche ist ein wichtiges Kriterium, da es sich stark auf die Akzeptanz und Nutzung der Funktion durch den Nutzer auswirkt. In diesem Kapitel wird deshalb darauf eingegangen, wie sich das alte Design zusammensetzte und welche Optimierungsmöglichkeiten bestehen. Aufbauend auf dieser Analyse wird eine neue Oberfläche entworfen, die die in DIN EN ISO 9241-110 „Grundsätze der Dialoggestaltung“[DIN 9241-110] aufgeführten und im Folgenden wiedergegebenen Aspekte berücksichtigt:

- **Aufgabenangemessenheit:** „Ein interaktives System ist aufgabenangemessen, wenn es den Benutzer unterstützt, seine Arbeitsaufgabe zu erledigen, d. h. wenn Funktionalität und Dialog auf den charakteristischen Eigenschaften der Arbeitsaufgabe basieren, anstatt auf der zur Arbeitserledigung eingesetzten Technologie.“
- **Selbstbeschreibungsfähigkeit:** „Ein Dialog ist in dem Maße selbstbeschreibungsfähig, in dem für den Benutzer zu jeder Zeit offensichtlich ist, in welchem Dialog, an welcher Stelle im Dialog er sich befindet, welche Handlungen unternommen werden können und wie diese ausgeführt werden können.“
- **Lernförderlichkeit:** „Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen der Nutzung des interaktiven Systems unterstützt und anleitet.“
- **Steuerbarkeit:** „Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.“
- **Erwartungskonformität:** „Ein Dialog ist erwartungskonform, wenn er den aus dem Nutzungskontext heraus vorhersehbaren Benutzerbelangen sowie allgemein anerkannten Konventionen entspricht.“
- **Individualisierbarkeit:** „Ein Dialog ist individualisierbar, wenn Benutzer die Mensch-System-Interaktion und die Darstellung von Informationen ändern können, um diese ab ihre individuellen Fähigkeiten und Bedürfnisse anzupassen.“
- **Fehlertoleranz:** „Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann. Fehlertoleranz wird mit den Mitteln erreicht:
  - Fehlererkennung und -vermeidung (Schadensbegrenzung)
  - Fehlerkorrektur und

– Fehlermanagement, um mit Fehlern umzugehen, die sich ereignen.“

5.1. Analyse des alten Designs

Das alte Design ist vor allem geprägt durch den dicken schwarzen Rahmen, der das Fenster umrundet (vgl. Abbildung 3.1). Um die runden Ecken zu realisieren, ist dieser Rahmen aus mehreren Bildern zusammengesetzt. Der Inhalt des Rahmens unterteilt sich in zwei Bereiche: den statischen unteren Teil, der die Buttons und Icons und somit die Editierfunktionen enthält, sowie den oberen dynamischen Bereich, der die Ordnerstruktur enthält. Unterschreitet das Fenster eine gewisse Mindestbreite beziehungsweise -höhe, so werden für den Nutzer Scrollbalken sichtbar (vgl. Abb. 5.1).

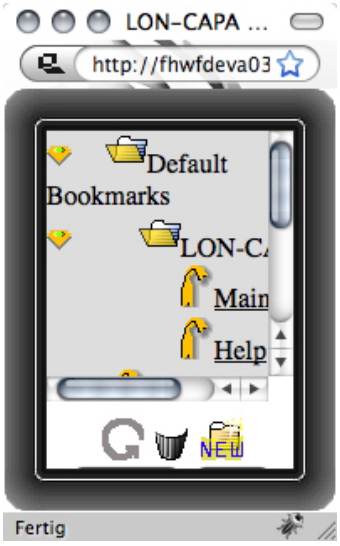

Abbildung 5.1.: Lesezeichen-Sammlung in LON-CAPA (altes Design) mit Scrollbalken

Diese Balken beziehen sich jedoch nur auf den oberen Bereich der Sammlung und nicht auf das gesamte Browser-Fenster. Dies hat zum Nachteil, dass eventuell nicht mehr sichtbare Elemente im unteren Bereich nicht mehr zugängig sind (vgl. fehlende Buttons in Abb. 5.1). Die Leisten, in denen die Scrollbalken erscheinen, befinden sich am rechten und unteren Rand des hellgrauen Bereiches und sind unabhängig von der Größe des Fensters immer sichtbar (vgl. Abbildung 3.1). Es wäre allerdings angebrachter, wenn diese dem Nutzer nur gezeigt würden, wenn es wirklich notwendig ist. Scrollbalken- und leisten sollten also nur dann sichtbar werden, wenn es die Fenstergröße bezogen auf den Fensterinhalt erfordert. Sie sollten sich dann außerdem auf das gesamte Fenster auswirken.
In der Visualisierung der Lesezeichen-Sammlung werden angelegte Ordner und Lesezeichen durch Icons unterschieden. Auch die Editierfunktionen „Neuen Ordner anlegen“ und „Löschen“ werden durch Icons abgebildet, während die anderen Funktionen durch Buttons dargestellt werden. Für den Nutzer wäre es jedoch verständlicher und einprägsamer,

wenn konsequent für alle Funktionen dieselbe Darstellungsweise benutzt würde. Hinzu kommt, dass die gewählten Icons nicht ideal sind. Das Mülleimer-Icon beispielsweise ist im ersten Moment nur schlecht als solches zu identifizieren.
Im Rahmen eines Redesign-Projektkurses wurde die gesamte Oberfläche von LON-CAPA, mit einigen Ausnahmen wie auch der Lesezeichen-Sammlung, überarbeitet und vereinheitlicht. Die Design-Elemente der Lesezeichen-Sammlung finden sich an keiner anderen Stelle im System wieder. Zur Steigerung der Lernförderlichkeit und um die Erwartungen des Nutzers, der an ein einheitliches Design gewöhnt ist, zu erfüllen, ist eine entsprechende Überarbeitung und Anpassung der Oberfläche also zwingend notwendig. Im folgenden Abschnitt wird deshalb auf die Neugestaltung der Dialoge eingegangen.

5.2. Gestaltung einer neuen Oberfläche

Ziel der Neugestaltung der Benutzeroberfläche ist es, eine ergonomische und intuitive Bedienung durch den Nutzer zu schaffen. Dazu ist es auf jeden Fall förderlich, eine Anpassung an das Design des restlichen System durch Nutzung bereits vorhandenen Strukturen und Styling-Elementen (zum Beispiel Wiederverwendung von Icons) vorzunehmen. Abbildung 5.2 zeigt, wie die Wunschliste durch Nutzung bereits vorhandener Elemente gestaltet werden kann.

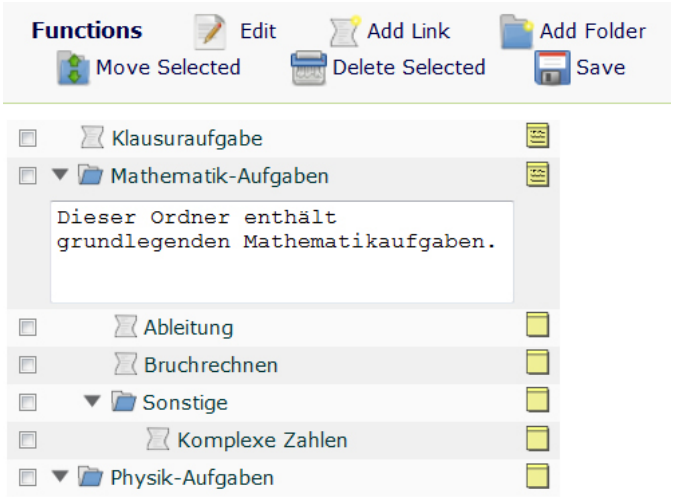

Abbildung 5.2.: Neugestaltung unter Verwendung bereits existenter Strukturen

Für die Strukturierung der anzuzeigenden Daten ist eine tabellarische Darstellung am geeignetsten. Für die Gestaltung von Tabellen gibt es in LON-CAPA einheitliche Styles. Die erste Zeile einer Tabelle, die Header-Zeile, wird dabei farblich hinterlegt. Standardmäßig ist der Farbton durch die Benutzerrolle vordefiniert, kann aber in den Benutzereinstellungen geändert werden. Die Zeilen der Tabelle werden abwechselnd mit zwei, nicht durch den Nutzer veränderbaren, verschiedenen Grautönen hinterlegt. Dies erleich-

tert dem Nutzer die Orientierung, in welcher Zeile er sich befindet. Jede Zeile enthält einen Eintrag der Wunschliste, wobei jeweils durch ein Icon symbolisiert wird, ob es sich um einen Ordner oder einen Link handelt. Zu jedem Ordner wird zudem ein Pfeil eingeblendet, über welchen der Ordner auf- beziehungsweise zugeklappt werden kann. Außerdem befindet sich in jeder Zeile eine Check-Box, über die zu verschiebende oder löschende Einträge ausgewählt werden können. Bezogen auf die Erwartungskonformität eines Dialoges befinden sich diese üblicherweise vor dem Text, auf den sie sich beziehen, um die Zugehörigkeit zum Text sowie eine gleichmäßige vertikale Anordnung aller sich untereinander befindender Boxen zu gewährleisten. Das letzte Element jeder Zeile ist ein Notiz-Icon. Über diese Icons können Notizen erstellt oder geändert werden. Bei einem Eintrag mit vorhandener Notiz zeigt das Icon einen beschriebenen Notiz-Zettel. Ist für einen Eintrag keine Notiz hinterlegt, so wird dies durch einen leeren Notiz-Zettel symbolisiert. Die Notizen sind standardmäßig ausgeblendet, um einen überladen aussehenden Bildschirmes zu verhindern. Über einen Klick auf das Notiz-Icon wird die zu dem Eintrag gehörende Notiz in einer Zeile unterhalb des Eintrages eingeblendet. Ein erneuter Klick auf das Icon blendet die entsprechende Notiz wieder aus.
Außerhalb der Tabelle befinden sich Icons, die Funktionen zum Editieren der Wunschliste anbieten. Die Icons „Neues Verzeichnis“ und „Neuer Link“ öffnen Fenster, in denen die Daten für ein neu anzulegendes Verzeichnis oder einen neu anzulegenden Link eingegeben werden können. Beim Anlegen kann dabei direkt entscheiden werden, in welches bereits existierende Verzeichnis der neue Eintrag gespeichert werden soll (vgl. Abb. 5.3).

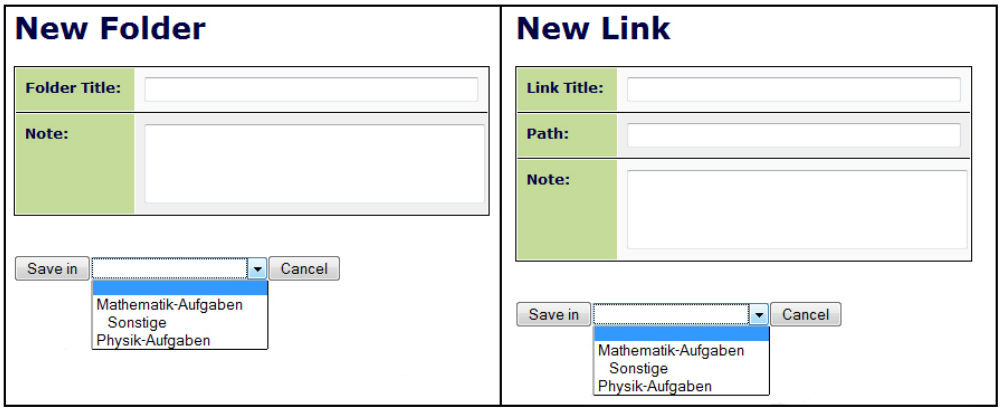

Abbildung 5.3.: Dialog zum Anlegen von neuen Ordner und Links

Soll der Titel oder Pfad eines Eintrages nachträglich geändert werden, so muss über ein entsprechendes Icon in den Editier-Modus gewechselt werden (siehe Abbildung 5.4). In diesem Modus stehen die Titel und Pfade in Textboxen und können dadurch verändert werden. Auch das Ändern oder Erstellen von Notizen ist in diesem Modus möglich. Zusätzlich werden im Editier-Modus Elemente eingeblendet, die das Sortieren von Einträgen ermöglichen. Da immer nur innerhalb eines Ordners sortiert werden kann, und um das Design übersichtlich zu halten, werden diese Elemente immer nur auf der aktu-

ell tiefsten Ebene eingeblendet. Um auf einer höheren Ebene zu sortieren, müssen alle Ordner tieferer Ebenen geschlossen werden. Damit die Änderungen an Titel, Pfad oder Notiz übernommen werden, müssen diese explizit gespeichert werden.

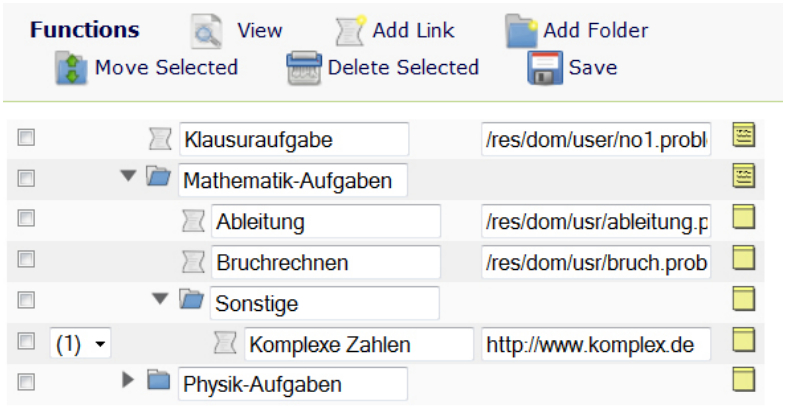

Abbildung 5.4.: Editier-Modus der Wunschliste

Das Gegenstück zum Editier-Modus ist der Betrachten-Modus. Die Ansicht entspricht der in Abbildung 5.2 veranschaulichten. Bei einem Klick auf den Titel eines Links wird dieser geöffnet.
Einträge werden verschoben oder gelöscht, indem der Nutzer sie über die zur Verfügung stehenden Check-Boxen auswählt und dann auf das zur Aktion passende Icon klickt. Diese Aktionen können in beiden Modi angewendet werden. Für das Verschieben von Einträgen wird nach der Auswahl der Einträge eine weitere Ansicht benötigt, in der der Nutzer den Zielort, an den verschoben werden soll, angeben kann. Dazu wird die gesamte Verzeichnisstruktur angezeigt und für jeden Ordner ein Radio-Button erzeugt. Über diesen wird der gewünschte Zielordner ausgewählt.

5.3. Bedienbarkeit

Ein wichtiger Aspekt bei der Gestaltung einer Benutzeroberfläche ist das Vermeiden beziehungsweise Abfangen und Korrigieren von Fehlverhalten des Nutzers (Fehlertoleranz). Im Folgenden wird erläutert, wie dieser Aspekt in der Wunschliste umgesetzt wird.
Das Löschen oder Verschieben eines Ordners wirkt sich immer auch auf den gesamten Ordnerinhalt aus. Ausgewählt werden die zu verschiebenden oder löschenden Einträge über Checkboxen. Dabei werden bei der Auswahl einer Checkbox, die zu einem Ordner gehört, automatisch alle Checkboxen, die zu den Einträgen innerhalb des ausgewählten Ordners gehören, ebenfalls ausgewählt. Sobald ein Eintrag innerhalb eines Ordners abgewählt wird, werden alle darüber liegenden Ordner ebenfalls abgewählt. Dadurch wird gewährleistet, dass der Nutzer beim Löschen oder Verschieben eines Ordners immer auch den Inhalt ausgewählt hat. Soll ein Eintrag eines Ordners nicht gelöscht oder verschobe-

ne werden, so können auch der entsprechenden Ordner und die übergeordneten Ordner nicht gelöscht oder verschoben werden. Alle anderen Link oder auch Ordner, die sich auf derselben Ebene wie der ausgewählte Eintrag befinden, dürfen jedoch gelöscht werden. Dadurch wird für eine konsistente Datenhaltung gesorgt. Der Nutzer wird zudem beim Löschen darauf hingewiesen, dass er beim Löschen eines Ordners auch den Inhalt dieses Ordners löscht und kann die Aktion bei Bedarf noch abbrechen.
Beim Verschieben von Ordnern ist zu bedenken, dass diese nicht in sich selbst oder in ihre eigenen Unterordner verschoben werden dürfen. Sobald ein Ordner zum Verschieben ausgewählt wurde, werden bei der Auswahl des Zielordners keine Radio-Buttons für den entsprechenden Ordner und seine Unterordner angezeigt, sondern nur für alle anderen sich in der Liste befindlichen Ordner.
Änderungen an Titel oder Notiz eines Eintrages benötigen ein explizites Speichern. Damit verhindert wird, dass eingegebene aber noch nicht gesicherte Änderungen verloren gehen, wird der Nutzer beim Verlassen der Seite auf noch nicht gespeicherte Veränderungen hingewiesen. Er kann dann entscheiden, ob er die Änderungen speichern oder verwerfen möchte.
Bei der Gestaltung der Wunschliste wurde darauf geachtet, eine möglichst intuitive Bedienung zu schaffen. Sollte ein Nutzer dennoch Anleitung benötigen, so wird für diesen Fall eine Hilfe angeboten, die die Intuition und Bedienung der Wunschliste erklärt und somit die Lernförderlichkeit unterstützt.

6. Implementierung

Das folgende Kapitel beschäftigt sich mit der Umsetzung der Wunschliste in das LON-CAPA-System. Es werden die Vorgehensweise und die eingesetzten Techniken erläutert.

6.1. Technische Aspekte

Das LON-CAPA-System besteht hauptsächlich aus Skripten der Programmiersprache Perl. Die Wunschliste wird ebenfalls in Perl umgesetzt und stellt ein eigenes Perl-Modul dar. Die in Kapitel 4 vorgestellte „Tree“-Klasse ist bereits in dem CPAN (engl. „Comprehensive Perl Archive Network“)-Modul „Tree“ implementiert und wird in dem Modul der Wunschliste verwendet. Zudem werden andere LON-CAPA-Module eingebunden, um bereits existierende Routinen verwenden zu können.
Zu diesen zählen die ebenfalls in Kapitel 4 vorgestellten Routinen „put“ und „dump“. Das Wunschlisten-Modul sorgt für die Speicherung und das Lesen der Wunschlisten-Daten unter Verwendung dieser Routinen. Außerdem wird HTML-Code für die Benutzeroberfläche erzeugt und die Benutzereingaben und -interaktionen verarbeitet. Bei der Erstellung des HTML-Codes ist teilweise ebenfalls eine Wiederverwendung bestehender Routinen möglich. Beispielsweise können Tabellen zentral unter Verwendung verfügbarer CCS (engl. „Cascading Style Sheets“)-Definitionen erstellt werden. Dies bietet den Vorteil, dass sich eine Änderungen, beispielsweise der Style-Definitionen, für alle Tabellen dieser Art auswirken und somit ein konsistentes Design beibehalten wird. Die erstellten HTML-Seiten sind XHTML 1.1 transitional validiert.
Das LON-CAPA-System ist auf eine weltweite Nutzung ausgelegt und bietet somit mehrere Oberflächensprachen an. Dazu werden die Texte in englischer Sprache einer zentralen Übersetzungs-Routine übergeben. Sind die entsprechenden Texte in der zugehörigen Sprachdatei hinterlegt, so übersetzt diese Routine automatisch in die vom Nutzer gewählte Oberflächensprache. Texte, die nicht in der Sprachdatei hinterlegt sind, werden in englischer Sprache angezeigt.
Die Wunschliste ist in einer Ordnerstruktur organisiert. Der Nutzer kann durch auf- und zuklappen der Ordner in dieser navigieren. Beim Laden der Seite wird die komplette Wunschliste geladen, jedoch nur die oberste Ebene angezeigt. Alle tiefer liegenden Ebenen sind ausgeblendet und können über Ordneraktionen ein- beziehungsweise auch wieder ausgeblendet werden. Dieses Ein- und Ausblenden wird über JavaScript realisiert und geschieht somit clientseitig. Dies bietet den Vorteil, dass nicht bei jedem Öffnen oder Schließen eines Ordners die komplette Seite neu geladen wird. Auch das Ein- und Ausblenden von Notizen wird auf diese Weise umgesetzt.
Die Dokumentation des Perl-Modul „Wunschliste“ wird mit Hilfe der Markup-Sprache POD (engl. „Plain Old Documentation“) realisiert. Diese ermöglicht eine Ausgabe in

verschieden Formate wie beispielsweise PDF oder HTML. Zusätzlich zu POD existieren zu jeder implementierten Routinen Perl-Kommentaren, die die Funktionen beschreiben. Dadurch wird den Entwicklern des Systems bei späterer Pflege, Überarbeitung oder Erweiterung des Moduls die Einarbeitung erleichtert. Die POD zu den implementierten Routinen ist in Anhang A zu finden.

6.2. Testfälle

Zur Feststellung der korrekten Funktionsweise des Wunschlisten-Modul sind Testfälle notwendig, die sämtliche Nutzer-Interaktivitäten (so genannte „Use Cases“) auflisten und diese auf die Reaktion des Systems prüfen. Diese Auflistung befindet sich im Anhang B „Use Cases“ der Arbeit. Alle Testfälle wurden durchgeführt und die Funktionalität der Wunschliste als so gut wie fehlerfrei bewertet. Nur in Ausnahmefällen kann inkonsequentes Nutzerverhalten zu unvorhersehbaren Ergebnissen führen. Die Auswertungen der Testfälle können ebenfalls den „Use Cases“ entnommen werden.

7. Zusammenfassung

Mit der Wunschliste wurde ein Modul geschaffen, welches dem Nutzer eine einfache und leicht zu bedienende Ordnungsstruktur auf für ihn relevante Teile des Ressourcenpools und externe Webseiten ermöglicht. In der Wunschliste hinterlegte Links können mit Hilfe von Ordnern strukturiert und durch Notizen annotiert werden. Funktionen wie das Verschieben und Sortieren von Links und Ordnern sowie die Möglichkeit des nachträglichen Änderns von Titeln, Pfaden und Notizen gestatten dem Nutzer zu jeder Zeit eine individuelle Anpassung seiner Wunschliste an seine Bedürfnisse. Das Design der Oberfläche, welches konsequent zum restlichen Aussehen des Systems gestaltet ist, unterstützt diese Tatsache.

7.1. Ausblick

Im Rahmen der Implementierung wurden bei zwei Funktionen Einschränkungen getroffen, da deren optimale Umsetzung Lösungen erfordert, die nicht nur für die Wunschliste sondern auch für andere Komponenten in LON-CAPA relevant sind. Zum einen betrifft dies die Platzierung des Icons, welches ein automatisches Setzen eines Links auf die Wunschliste für eine Ressource aus der Suche beziehungsweise der „Veröffentlichte Ressourcen“-Ansicht heraus ermöglicht. Dieses Icon wird momentan direkt hinter dem jeweiligen Titel einer Ressource eingeblendet. Besser wäre jedoch, dieses in der Vorschau einer Ressource anzuzeigen. Momentan gibt es in dieser Ressourcen-Vorschau keinen Header, in dem Funktionen, die auf die Ressource ausgeführt werden können, gesammelt werden könnten. Zu diesen Funktionen zählt neben der Möglichkeit, einen Link auf die Wunschliste zu setzen, in jedem Fall auch das Drucken einer Ressource. Es ist also ratsam, für eben solche Funktionen ein Gruppierelement zu schaffen, wie es bereits auch an anderen Stellen in ähnlicher Weise existiert. Dort werden dann sämtliche Funktionen, die auf Ressourcen im Vorschau-Modus angewendet werden können, aufgelistet.
Die andere Einschränkung bezieht sich auf die Überprüfung eines Links hinsichtlich seiner Kategorie (vgl. Kapitel 3.2). Derzeit wird geprüft, ob der eingegeben Pfad, sollte er nicht mit /res/... beginnen, einen LON-CAPA-„Datentyp“ (wie beispielsweise .problem, .exam oder .quiz) enthält. In diesem Fall wird davon ausgegangen, dass es sich um einen Pfad auf den Inhalt eines LON-CAPA-Kurses handelt (vgl. Linktyp Kategorie 3 in Kapitel 3.2) und das Anlegen eines solchen Links in der Wunschliste nicht gestattet. Dies führt jedoch dazu, dass alle externen Seiten, die zufällig einen LON-CAPA-„Datentyp“ in ihrem Pfad enthalten, nicht in der Wunschliste eingetragen werden können. Derselbe Prüf-Mechanismus wird auch beim Anzeigen von in den Kurs eingebundenen „externen Ressourcen“ verwendet. Wird eine Webseite eingebettet, die einen LON-CAPA-„Datentyp“ in ihrem Pfad enthält, so führt der beschrieben Prüf-Mechanismus dazu, dass diese

Seite nicht angezeigt wird. Es bedarf also einer grundsätzlichen Lösung, um für einen gegeben Pfad überprüfen zu können, ob dieser zu einem LON-CAPA-Server führt oder nicht.
Die Wunschliste bietet zudem Erweiterungsmöglichkeiten (vgl. Kapitel 3.3) welche im Rahmen der stetigen Weiterentwicklung des Systems realisiert werden können. Bei der ständigen Optimierung und weiteren Vereinheitlichung muss auch die Wunschliste bedacht werden. Die einfache Strukturierung und die Dokumentation des Perl-Moduls erleichtern dabei die Arbeit.
Die in der Wunschliste hinterlegten Links können dazu beitragen, die Suchfunktionalität im Ressourcenpool zu verbessern. So wäre es beispielsweise denkbar, aufgrund von bereits verlinkten Ressourcen eines Nutzer, diesem Nutzer bei der Suche bestimmte Ressourcen präferiert anzuzeigen. Weiterhin können, aufbauend auf den Links in den Wunschlisten, Netze zwischen Nutzern geschaffen werden. Nutzer, die vermehrt die selben Ressourcen verlinken, haben dabei eine stärkere Verbindung also solche, die keine Ressourcen-Links teilen. Diese Vernetzungen können ebenfalls die Suchergebnisse optimieren. Die Implementierung der bereits in Kapitel 3.3 erläuterten Import- und Export-Funktionen wären dabei für den schnellen Austausch von Wunschlisten zwischen verschiedenen Nutzern hilfreich. Durch die genannten Punkte würde der Netzwerk-Aspekt des LON-CAPA-Systems weiter gestärkt.
Eine weitere Ausbaumöglichkeiten um einen der Grundaspekte LON-CAPA's, nämlich die Wiederverwendung von Ressourcen, zu stärken, bestände darin, eine Funktion anzubieten, die das Kopieren von Ordner der Wunschliste in den Konstruktionsbereich erlaubt. Dort könnten diese, wie auch Kursinhalte, die in den Konstruktionsbereich geladen werden können, bearbeitet und zur Wiederverwendung veröffentlicht werden.
Weiterhin wäre zu durchdenken, ob die Wunschliste auch für Studenten zugängig sein sollte. Studenten könnten darauf Links zu externen Seiten und zu Ressourcen, die sich in einem Kurs befinden, speichern. Der Link für Ressourcen würde dann auf den Ressourcenpool zeigen. Ist die Ressource zu einem Link in dem derzeit ausgewählten Kurs eingebunden, so erhält der Student beim Betrachten des Link diese kursgebundene Version der Ressource, auf die er ohnehin Zugriff hat. Sollte die zu betrachtende Ressource nicht im Kurs eingebunden sein, so würde er eine Meldung über fehlende Zugriffsrechte erhalten, da ein Student keinen Zugriff auf den Ressourcenpool hat. Es ist jedoch vorstellbar, dass ein Nutzer mit vorerst studentischer Roll zusätzlich eine Rolle zugewiesen bekommt, die den Zugriff auf den Ressourcenpool ermöglicht. In dieser Rolle könnte er dann, sofern die Zugriffrechte auf die Ressourcen nicht durch den Autor beschränkt wurden, sämtliche in der Wunschliste hinterlegten Ressourcen-Link betrachten. Zusätzlich erhält er dann auch die Möglichkeit, seine Links in den entsprechenden Kurs zu importieren.

Literaturverzeichnis

[DIN 9241-110] DIN EN ISO 9241-110 „Grundsätze der Dialoggestaltung“, Stand: September 2008

[EMC] http://germany.emc.com/about/news/press/2010/20100504-01.htm (letzter Abruf am 18.08.2010)

[Bugzilla] http://bugs.loncapa.org Bug 1810 und Bug 1892 (letzter Abruf am 18.08.2010)

[LON-CAPA] http://www.lon-capa.org (letzter Abruf am 18.08.2010)

Anhang

A. Implementierte Routinen

Wishlist-Module

The wishlist offers a possibility to store links to resources from the resource-pool and external websites in a hierarchical list. It is only available for user with access to the resource-pool. The list can be structured by folders.
The wishlist-module uses the CPAN-module „Tree“ for easily handling the directory-structure of the wishlist. Each node in the tree has an index to be referenced by.

Routines for getting and putting the wishlist data from and accordingly to users data

- &getWishlist()
  Get the wishlist-data via lonnet::dump() and returns the got data in a hash.
- &putWishlist(wishlist)
  Parameter is a reference to a hash. Puts the wishlist-data contained in the given hash via lonnet::put() to user-data.
- &deleteWishlist()
  Deletes all entries from the user-data for wishlist. Do this before putting in new data.

Routines for changing the directory struture of the wishlist.

- &newEntry(title, path, note)
  Creates a new entry in the wishlist containing the given informations. Additionally saves the date of creation in the entry.
- &deleteEntries(marked)
  Parameter is a reference to an array containing the indices of all nodes that should be removed from the tree.
- &sortEntries(indexNode, at)
  Changes the position of a node given by indexNode within its siblings. New position is given by at.
- &moveEntries(indexNodesToMove, indexParent)
  Parameter is a reference to an array containing the indices of all nodes that should

be moved. indexParent specifies the node that will become the new Parent for these nodes.

- &setNewTitle(nodeindex, newTitle)
  Sets the title for the node given by nodeindex to newTitle.
- &setNewPath(nodeindex, newPath)
  Sets the path for the node given by nodeindex to newPath.
- &setNewNote(nodeindex, newNote)
  Sets the note for the node given by nodeindex to newNote.
- &saveChanges()
  Prepares the wishlist-hash to save it via &putWishlist(wishlist).

Routines for handling the directory structure

- &getFoldersForOption(nodes)
  Return the titles for all exiting folders in an option-tag, used to offer the users a possibility to create a new link or folder in an existing folder. Recursive call starting with all children of the root of the tree (parameter nodes is reference to an array containing the nodes of the current level).
- &getfoldersOption()
  Returns the option-tag build by &getFoldersForOption(nodes). Use it to transfer this to other modules (e.g. lonmenu.pm).
- &getFoldersToArray(children)
  Puts all nodes that represent folders in the wishlist into an array. Recursive call starting with all children of the root of the tree (parameter nodes is reference to an array containing the nodes of the current level).
- &getNodesToArray(children)
  Puts all existing nodes into an array (apart from the root node, because this one does not represent an entry in the wishlist). Recursive call starting with all children of the root of the tree (parameter nodes is reference to an array containing the nodes of the current level).

Routines for the user-interface of the wishlist

- &JSforWishlist()
  Returns JavaScript-functions needed for wishlist actions like open and close folders.
- &wishlistView(nodes)
  Returns the table-HTML-markup for the wishlist in mode „view„. Recursive call starting with all children of the root of the tree (parameter nodes is reference to an array containing the nodes of the current level).

- &wishlistEdit(nodes)
  Returns the table-HTML-markup for the wishlist in mode „edit„. Recursive call starting with all children of the root of the tree (parameter nodes is reference to an array containing the nodes of the current level).

- &wishlistMove(nodes, marked)
  Returns the table-HTML-markup for the wishlist in mode „move„. Highlights all entry „selected to move„ contained in marked (reference to array). Recursive call starting with all children of the root of the tree (parameter nodes is reference to an array containing the nodes of the current level).

- &wishlistImport(nodes)
  Returns the table-HTML-markup for the wishlist in mode „import„. Recursive call starting with all children of the root of the tree (parameter nodes is reference to an array containing the nodes of the current level).

- &makePage(mode, marked)
  Returns the HTML-markup for the whole wishlist depending on mode. If mode is „move„ we need the marked entries to be highlighted a „selected to move„. Calls &wishlistView(nodes), &wishlistEdit(nodes) or &wishlistMove(nodes, marked).

- &makePageSet()
  Returns the HTML-Markup for the page shown when a link was set by using the icon when viewing a resource.

- &makePageImport()
  Returns the HTML-Markup for the page shown when links should be imported into courses.

- &makeErrorPage ()
  Returns the HTML-Markup for an error-page shown if the wishlist could not be loaded.

Routines from package Tree

- &getNodeByIndex(index, nodes)
  Searches for a node, specified by the index, in nodes (reference to array) and returns it.

- &moveNode(node, at, newParent)
  Moves a given node to a new parent (if new parents is defined) or change the position from a node within its siblings (means sorting, at must be defined).

- &removeNode(node)
  Removes a node given by node from the tree.

- &TreeIndex(children)
  Sets an index for every node in the tree, beginning with 0. Recursive call starting with all children of the root of the tree (parameter children is reference to an array containing the nodes of the current level).

- &setCountZero()
  Resets index counter.

- &RootToHash(childrenRt)
  Converts the root-node to a hash-entry: the key is root and values are just the indices of root's children.

- &TreeToHash(childrenRt)
  Converts all other nodes in the tree to hash. Each node is one hash-entry where the keys are the index of a node and the values are all other attributes (containing tile, path, note, date and indices for all direct children). Recursive call starting with all children of the root of the tree (parameter childrenRT is reference to an array containing the nodes of the current level).

- &HashToTree()
  Converts the hash to a tree. Builds a tree-object for each entry in the hash. Afterwards call &buildTree(node, childrenIn, TreeNodes, TreeHash) to connect the tree-objects.

- &buildTree(node, childrenIn, TreeNodes, TreeHash)
  Joins the nodes to a tree. Recursive call starting with root and all children of root (parameter childrenIn is reference to an array containing the nodes indices of the current level).

B. Use Cases

Jeder Use-Case hat zur Vorbedingung, dass der Nutzer im LON-CAPA-Netzwerk eingeloggt sein muss. Im Weiteren werden nur zusätzliche Vorbedingungen erwähnt. Außerdem kann es bei jedem Use-Case zu Serverproblemen oder Netzwerkfehlern kommen. Diese sind also in jedem Use-Case als Fehlerfall zu betrachten und werden nicht jedes mal explizit erwähnt.

Wunschliste betrachten

Vorbedingung:	-
Beschreibung:	Der Nutzer betrachtet seine bisherige Wunschliste. Dies kann er sowohl im Editier- als auch im Betrachten-Modus tun.
Nachbedingung:	-
Fehlerfälle:	-
Testergebnis:	-

Neuen Ordner erstellen

Vorbedingung:	Die Wunschliste muss aufgerufen sein.
Beschreibung:	Der Nutzer legt einen neuen Ordner in seiner Liste an. Dazu wählt er einen Ordner-Titel und gegebenenfalls einen Zielordner, als dessen Unterordner der neu anzulegende Ordner erzeugt werden soll. Außerdem kann dem anzulegenden Ordner eine Notiz hinzugefügt werden.
Nachbedingung:	Ein neuer Ordner wurde erzeugt und befindet sich entweder in dem gewählten Zielordner oder, wurde dieser nicht gewählt, auf der obersten Verzeichnisebene.
Fehlerfälle:	Unzulässige Zeichen im Ordner-Titel.
Testergebnis:	Sämtliche Zeichen sind zulässig, somit sind jegliche Ordner-Titel legitim.

Ordner umbenennen

Vorbedingung:	Die Wunschliste muss im Editier-Modus aufgerufen sein und es muss ein Ordner existieren, der umbenannt werden soll.
Beschreibung:	Der Nutzer nennt einen Ordner um. Dazu ändert er den Ordner-Titel in dem dafür vorgesehenen Eingabefeld.
Nachbedingung:	Der ausgewählte Ordner besitzt nun den neuen Ordner-Titel.
Fehlerfälle:	Unzulässige Zeichen im Ordner-Titel.
Testergebnis:	Sämtliche Zeichen sind zulässig, somit sind jegliche Ordner-Titel legitim.

Ordner verschieben

Vorbedingung:	Die Wunschliste muss aufgerufen sein und es muss ein Ordner existieren, der verschoben werden soll.
Beschreibung:	Der Nutzer verschiebt einen Ordner. Dazu wählt er zuerst einen Ordner und dann das neue Ziel-Verzeichnis aus. Dieses Ziel-Verzeichnis kann auch die oberste Verzeichnis-Ebene der Wunschliste sein.
Nachbedingung:	Der Ordner befindet sich nun in dem neuen Verzeichnis.
Fehlerfälle:	Der ausgewählte Ordner oder der ausgewählte Zielordner existiert nicht mehr (zwischenzeitlich gelöscht).
Testergebnis:	Beim Verschieben ist es wichtig, dass der Nutzer den Vorgang zu Ende ausführt, bevor er eine neue Aktion beginnt. Das Verschieben von Ordnern kann als Verschieben von Positionen betrachtet werden. Sollte sich der Inhalt einer Position durch inkonsequentes Nutzerverhalten (Löschen eines Ordners obwohl dieser gerade verschoben werden soll) ändern, so werden dennoch die gewählten Positionen (mit eventuell geändertem Inhalt) verschoben.

Ordner löschen

Vorbedingung:	Die Wunschliste muss aufgerufen sein und es muss ein Ordner existieren, der gelöscht werden soll.
Beschreibung:	Der Nutzer wählt mindestens einen Ordner aus, um diesen zu löschen. Beim Löschen eines Ordners wird auch dessen Inhalt gelöscht. Der Nutzer muss den Löschvorgang explizit bestätigen.
Nachbedingung:	Der Ordner samt Inhalt ist gelöscht.
Fehlerfälle:	Der ausgewählte Ordner existiert nicht mehr (zwischenzeitlich gelöscht).
Testergebnis:	Die Aktion „Löschen“ kann nicht unterbrochen, sondern nur abgebrochen werden. Der beschriebene Fehlerfall kann also nicht eintreten.

Neuen Link erstellen

Vorbedingung:	Die Wunschliste muss aufgerufen sein.
Beschreibung:	Der Nutzer legt einen neuen Link an. Dazu gibt er einen Titel und einen Pfad sowie gegebenenfalls eine Notiz ein und wählt den Ziel-Ordner aus, in dem der Link abgelegt werden soll. Dieser Ziel-Ordner kann auch die oberste Verzeichnis-Ebene der Wunschliste sein. Es wird geprüft, ob der eingegebene Pfad zu einer LON-CAPA-Ressource oder einer externen Webseite gehört.
Nachbedingung:	Es wurde ein neuer Link mit dem gewählten Titel und dem gewählten Pfad erstellt. Außerdem wurde der Link in dem gewählten Ordner hinterlegt und die eingegebene Notiz hinzugefügt.
Fehlerfälle:	Der eingegebene Pfad gehört nicht zu einer LON-CAPA-Ressource oder einer externen Webseite. Unzulässige Zeichen im Titel, dem Pfad oder der Notiz.
Testergebnis:	Sämtliche Zeichen sind zulässig, somit sind jegliche Titel, Pfade und Notizen legitim. Genügt der eingegeben Pfad nicht den Bedingungen, so erhält der Nutzer darüber eine Meldung und kann entweder einen neuen Pfad eingeben (der natürlich auch wieder geprüft wird) oder das Anlegen des Links abbrechen.

Link verschieben

Vorbedingung:	Die Wunschliste muss aufgerufen sein und es muss ein Link existieren, der verschoben werden soll.
Beschreibung:	Der Nutzer verschiebt einen Link in einen anderen Ordner. Dazu wählt er einen Ziel-Ordner, in dem der Link zukünftig hinterlegt sein soll. Dieser Ziel-Ordner kann auch die oberste Verzeichnis-Ebene der Wunschliste sein.
Nachbedingung:	Der ausgewählte Link befindet sich nun in dem neuen Ordner.
Fehlerfälle:	Der ausgewählte Zielordner oder der ausgewählte Link existiert nicht mehr (zwischenzeitlich gelöscht).
Testergebnis:	Beim Verschieben ist es wichtig, dass der Nutzer den Vorgang zu Ende ausführt, bevor er eine neue Aktion beginnt. Das Verschieben von Links kann als Verschieben von Positionen betrachtet werden. Sollte sich der Inhalt einer Position durch inkonsequentes Nutzerverhalten (Löschen eines Links obwohl dieser gerade verschoben werden soll) ändern, so werden dennoch die gewählten Positionen (mit eventuell geändertem Inhalt) verschoben.

Link-Titel bearbeiten

Vorbedingung:	Die Wunschliste muss aufgerufen sein und der Nutzer muss sich im Editier-Modus befinden.
Beschreibung:	Der Nutzer bearbeitet den Titel eines Link. Dazu ändert er den Titel in dem dafür vorgesehenen Eingabefeld.
Nachbedingung:	Der Link enthält jetzt den geänderten Titel.
Fehlerfälle:	Unzulässige Zeichen im Titel.
Testergebnis:	Sämtliche Zeichen sind zulässig, somit sind jegliche Titel legitim.

Link-Pfad bearbeiten

Vorbedingung:	Die Wunschliste muss aufgerufen sein und der Nutzer muss sich im Editier-Modus befinden.
Beschreibung:	Der Nutzer bearbeitet den Pfad eines Link. Dazu ändert er den Pfad in dem dafür vorgesehenen Eingabefeld.
Nachbedingung:	Der Link enthält jetzt den geänderten Pfad.
Fehlerfälle:	Unzulässige Zeichen im Pfad.
Testergebnis:	Sämtliche Zeichen sind zulässig, somit sind jegliche Pfade legitim.

Link löschen

Vorbedingung:	Die Wunschliste muss aufgerufen sein und es muss ein Link existieren, der gelöscht werden soll.
Beschreibung:	Der Nutzer wählt mindestens einen Link aus, der gelöscht werden soll.
Nachbedingung:	Die ausgewählten Links sind gelöscht.
Fehlerfälle:	Der ausgewählte Link existiert nicht mehr (zwischenzeitlich gelöscht).
Testergebnis:	Die Aktion „Löschen“ kann nicht unterbrochen, sondern nur abgebrochen werden. Der beschriebene Fehlerfall kann also nicht eintreten.

Link-Notiz betrachten

Vorbedingung:	Die Wunschliste muss aufgerufen sein und es muss ein Link existieren, dessen Notiz betrachtet werden sollen.
Beschreibung:	Der Nutzer sieht die Notiz eines Links ein, indem er das zu dem Link gehörige Notiz-Icon anklickt.
Nachbedingung:	-
Fehlerfälle:	-
Testergebnis:	-

Link-Notiz hinzufügen

Vorbedingung:	Es muss ein Link existieren, dem eine Notiz hinzufügt werden soll. Dieser Link kann auch gerade erzeugt werden.
Beschreibung:	Der Nutzer fügt dem Link eine Notiz hinzu. Dies kann er auch nachträglich, nachdem der Link angelegt wurde, tun. Dazu nutzt er das zu dem Link gehörende Notiz-Icon.
Nachbedingung:	Der Link ist nun mit der eingegebenen Notiz verknüpft.
Fehlerfälle:	Unzulässige Zeichen in der Notiz.
Testergebnis:	Sämtliche Zeichen sind zulässig, somit sind jegliche Notizen legitim.

Link(s) importieren (Kursimport)

Vorbedingung:	Die Wunschliste muss mindestens einen gültigen Eintrag enthalten.
Beschreibung:	Der Nutzer importiert Ressourcen über die Wunschliste in den Kurs. Dazu wählt er die zu importierenden Ressourcen aus.
Nachbedingung:	Die ausgewählten Ressourcen befinden sich im Kurs.
Fehlerfälle:	Mindestens einer der ausgewählten Links existiert nicht mehr (zwischenzeitlich gelöscht).
Testergebnis:	Beim Importieren werden alle Einträge der Wunschliste aufgelistet. Jeder Eintrag kann als Position betrachtet werden. Sollte der Nutzer den Import-Bildschirm aufgerufen und danach Links aus seiner Liste löschen oder verschieben, so rücken alle anderen Einträge an entsprechende andere Positionen. Der Import-Bildschirm wird jedoch nicht automatisch aktualisiert, so dass dies dazu führen kann, dass andere Einträge als die ausgewählten importiert werden. In diesem Fall müsste der Import wiederholt werden.

Link setzen (Ressourcenbetrachtung)

Vorbedingung:	Der Nutzer betrachtet eine Ressource.
Beschreibung:	Der Nutzer setzt ein Link zu der betrachteten Ressourcen. Die Felder „Titel“ und „Pfad“ sind bereits mit dem Titel und dem Pfad der Ressource ausgefüllt. Der Nutzer kann den Titel jedoch vor dem Speichern ändern und bei Bedarf eine Notiz hinzufügen.
Nachbedingung:	Es befindet sich ein neuer Link mit den angegebenen Details in der Wunschliste.
Fehlerfälle:	Unzulässige Zeichen im Namen, dem Pfad oder der Notiz.
Testergebnis:	Sämtliche Zeichen sind zulässig, somit sind jegliche Titel, Pfade und Notizen legitim.