Design-Beschreibung des Programms zur Verwaltung des Übungsbetriebs

Version	Autor	Graphiken	Datum
1.0	Heike Jänicke	Frank Eckhardt Daniel Müller	08. Juni 2003
1.1	überarbeitet von Nils Ritter, Frank Eckhardt		16.Juni 2003
1.2	überarbeitet von Frank Eckhardt		23.Juni 2003
1.3	überarbeitet von Frank Eckhardt		30.Juni 2003
2.0	überarbeitet von Frank Eckhardt		07.Juli 2003

1. Allgemeines

Das Übungsverwaltungsprogramm ist ein servergestütztes Java-Projekt, das der Verwaltung von Seminaren zu universitären Vorlesungen dient.
Das Programm unterstützt dabei sowohl Studenten, als auch Dozenten und Tutoren bei der Verwirklichung der ihnen in diesem Zusammenhang gestellten Aufgaben Um dies zu ermöglichen ist es nötig, das Programm an die Anforderungen der verschiedenen Rollen anzupassen. Dabei werden den oben aufgeführten Rollen grundlegend folgende Benutzungsmöglichkeiten bereitgestellt.

Jede der einzelnen Rollen kann seine persönlichen Daten ändern, Dokumente downloaden und das interne Forum nutzen. Dazu kommen dann noch die Rollenspezifischen Funktionen:

- Studenten
  - Punkteübersicht
  - Einreichen gelöster Übungsaufgaben
  - Einschreibung zu Klausuren
  - Anmeldung zu Seminaren

- Tutoren
  - Eintragen der erreichten Punkte in ein dafür vorgesehenes Formular
  - Herunterladen gelöster Aufgaben zur Korrektur

- Dozenten
  - Bereitstellung von Dokumenten (Übungsaufgaben, Musterlösungen, Übersichten, etc.)
  - Daten erfassen (Anzahl angemeldeter Studenten, Punkteverteilungen, etc.)
  - Gruppenverwaltung (Termin, Ort, max. Anzahl Teilnehme)

- Admin
  - Benutzer hinzufügen
  - Benutzer löschen
  - Konfigurieren (Mailserverdaten festlegen
  - Archivieren (Daten archivieren und wiederherstellen)

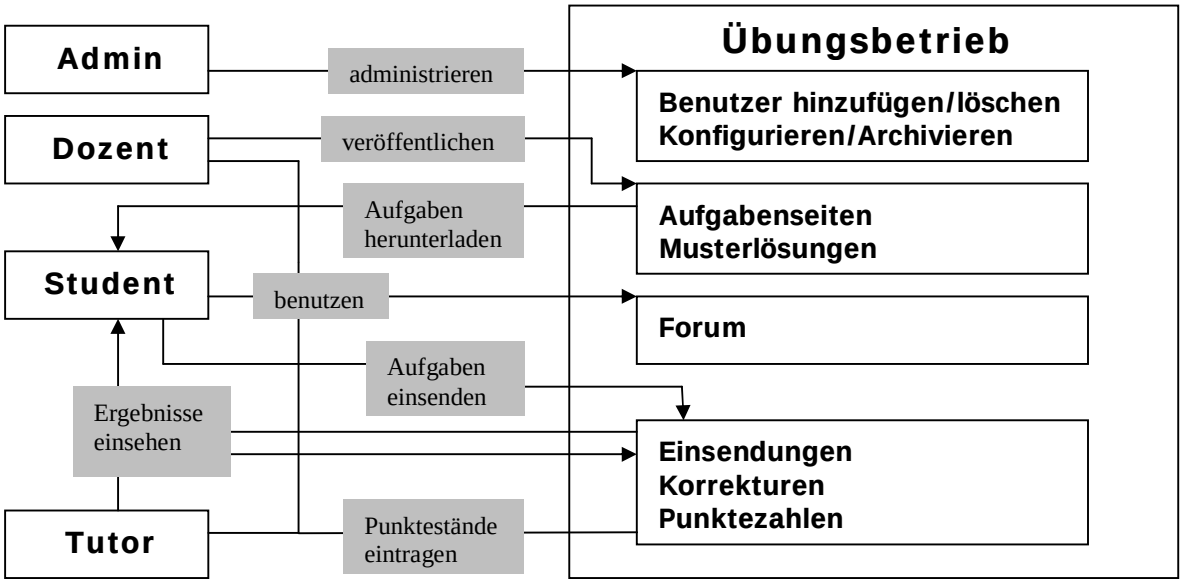

Abb. 1: Übersicht über die Hauptfunktionalitäten des Programms Übungsbetrieb

Realisiert wird das Projekt in Java mit Hilfe von Servlets (siehe 3.1 Servletarchitektur), welche den steten Datenaustausch ermöglichen. Zum Starten muss die Java JRE 1.4 bereitgestellt werden. Die benötigten Daten werden mit Hilfe von XML-Dateien gesichert, welche für den Umfang der zu verarbeitenden Daten angemessen sind.

2. Produktübersicht

Für alle Nutzer startet das Programm auf der Startseite mit dem Login. Hier wird er aufgefordert sowohl Namen, als auch Passwort anzugeben. Solange die Eingaben nicht korrekt sind, verbleibt der Anwender außerhalb des internen Bereichs und hat erneut die Möglichkeit Daten anzugeben. Sind diese korrekt, so wird er automatisch auf die interne Startseite weitergeleitet, wo sich automatisch das an seine Nutzungsmöglichkeiten angepasste Startmenü aufbaut. Von hier aus kann der Anwender mittels Anklicken der entsprechenden Schaltfläche zwischen den einzelnen Services wählen, die ihm zur Verfügung stehen. Das neue Servlet wird automatisch aufgerufen und der Benutzer gelangt auf eine neue Seite auf der er den gewünschten Service ausführen kann. Auch hier wird wieder das Menü eingeblendet, so dass er jederzeit die Möglichkeit hat, einen neuen Service auszuwählen oder sich auszuloggen.

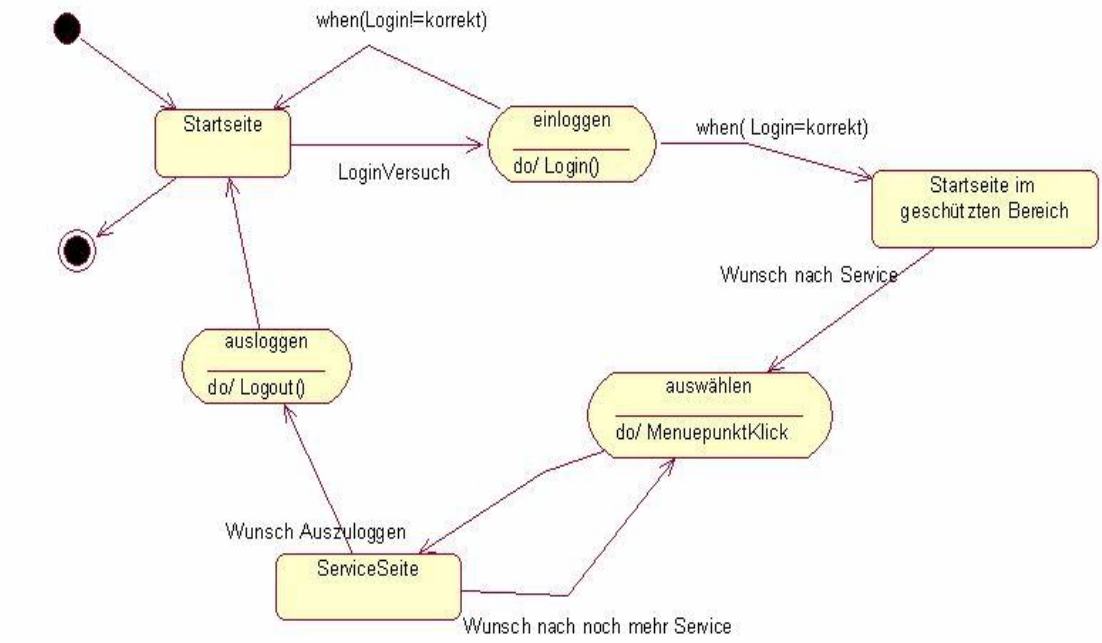

Abb. 2: Aktivitätsdiagramm des Projekts Übungsbetrieb

Da das Programm die gesamte Verwaltung des Übungsbetriebes unterstützen soll, ist es notwendig die Nutzer in verschiedene Klassen einzuteilen, um sicherzustellen, dass jeder Nutzer die seiner Rolle entsprechenden Services nutzen kann. Unterschieden werden beim Login die Rollen Student, Tutor und Dozent. Ihnen stehen die dem Diagramm zu entnehmenden Aktionen zur Verfügung.

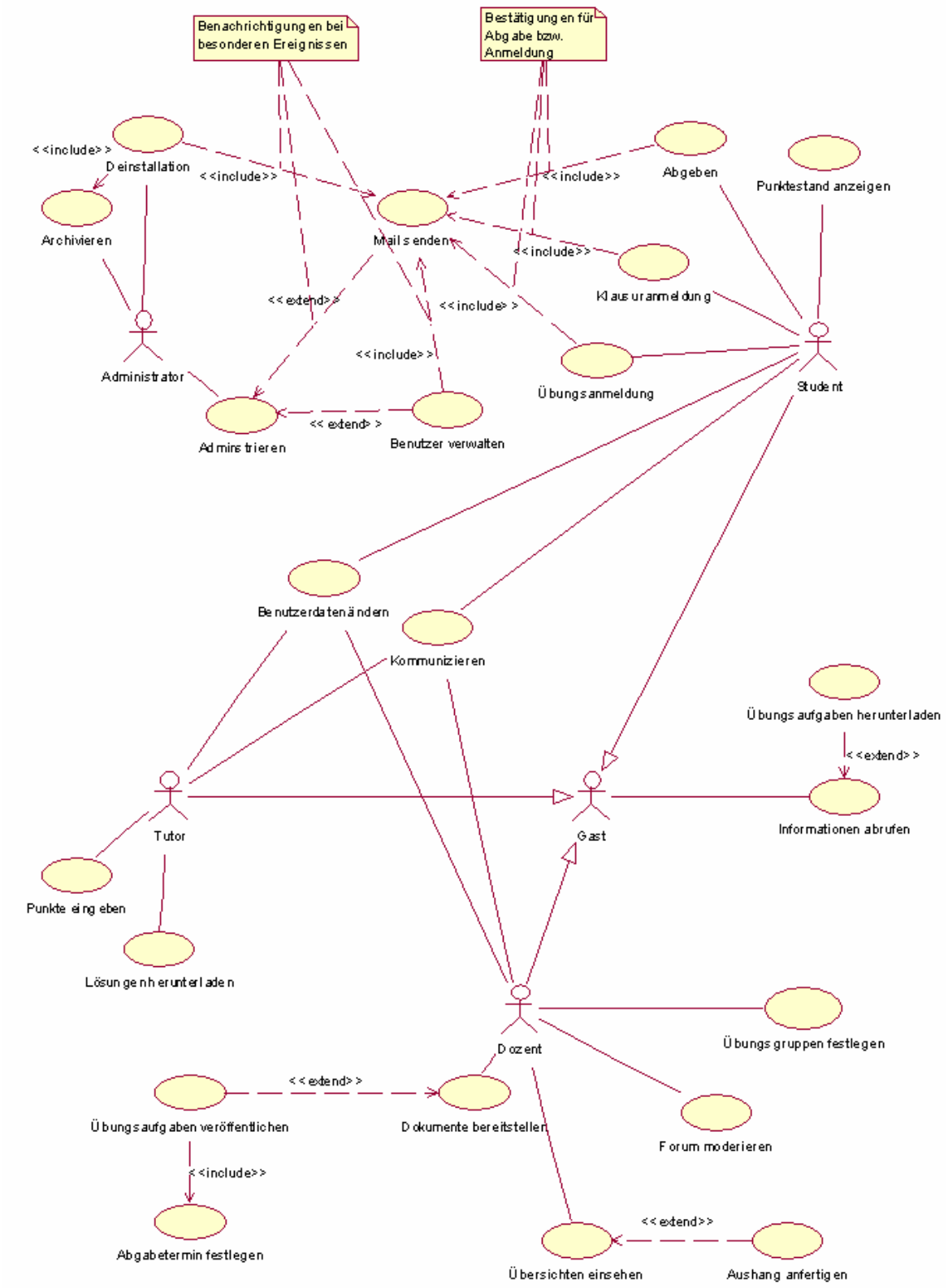

Abb. 3: Geschäftsprozessdiagramm des Projekts Übungsbetrieb

3. Grundsätzliche Designentscheidungen

3.1. Servlet-Architektur

Das Java-Projekt Übungsbetrieb wird mit Hilfe von Servlets realisiert. Der wichtigste Aspekt für die Verwendung dieses Konzepts ist die gute Unterstützung bei Web-Anwendungen. Dabei wird dem Entwickler ein einfacher, konsistenter Mechanismus bereitgestellt, der die Funktionalitäten des Web-Servers erweitert. Ein Servlet agiert im Grundlegenden wie ein Applet, mit dem Unterschied, dass es serverseitig gestartet wird und somit eine Vielzahl von Web-Applikationen erst möglich macht. Ferner sind Servlets sowohl server- als auch plattformunabhängig.

Im Falle von Übungsverwaltung werden Servlets benötigt, da ein steter Datenaustausch zwischen Benutzer und Server stattfindet, sich die anzuzeigenden Daten permanent ändern und das Programm auf eine Datenbank bzw. XML-Dateien zugreifen muss.

Grundsätzlich liegt allen Servlets folgende Struktur zugrunde:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;


public class SomeServlet extends HttpServlet {
   public void doGet(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException
   {  // Use "request" to read incoming HTTP headers (e.g. cookies)
      // and HTML form data (e.g. data the user entered and submitted)


      // Use "response" to specify the HTTP response line and headers
      // (e.g. specifying the content type, setting cookies).


   PrintWriter out = response.getWriter();
   // Use "out" to send content to browser
  }
}
```

3.2. Datenfluss

Gestartet wird das Programm mit dem Servlet Start. Dieses liest aus dem HttpServletRequest den eingegeben Benutzernamen und das Passwort aus. Diese werden durch das Servlet mit den in der user.xml vorhandenen Werten verglichen. Wird die Anmeldung nicht bestätigt, so wird dem Nutzer dies angezeigt und um eine erneute Eingabe gebeten. Andernfalls wird für den Anwender eine individuelle Session erzeugt. Anschließend wird die Welcome Klasse aufgerufen, von wo aus der Nutzer die jeweils gewünschten Aktionen aufrufen kann. Diese werden in dem Hauptmenü ausgewählt, und rufen die jeweilige Klasse aus dem Packet services auf.

Die nachfolgende Abbildung zeigt an, welche Daten von den einzelnen Seiten erhalten, erzeugt, bzw. weitergegeben werden und wie der Nutzer zwischen den einzelnen Zuständen wechseln kann.

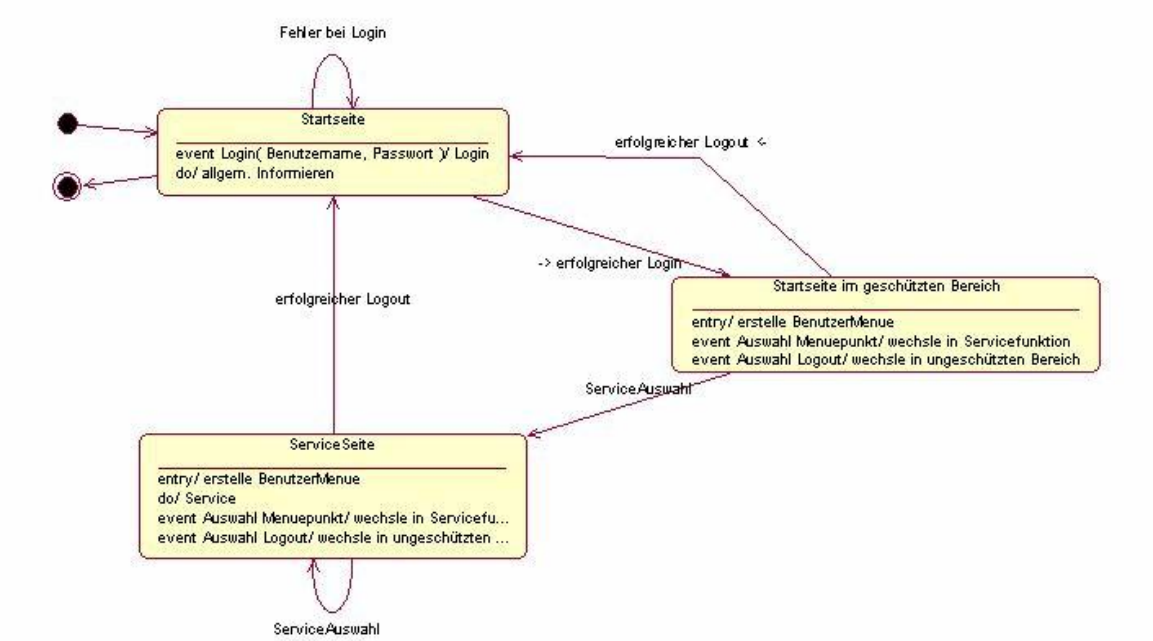

Abb. 4: Zustandsdiagramm des Projekts Übungsverwaltung

3.3. ...bei der Anmeldung

Ein wichtiger Aspekt bei der Sicherheit ist die Frage, wie mit dem Nutzer bzw. seiner Session zu verfahren ist, wenn er sich nicht richtig abmeldet (Schließen des Fensters ohne Logout) oder zu lange inaktiv ist. Das Problem liegt darin, dass der Server in ersterem Fall nicht weiß, dass der User nicht mehr im System ist und somit eine erneute Anmeldung verweigern würde. Mögliche Lösungsstrategien waren:

a) Der Benutzer kann sich für einen bestimmten Zeitraum nicht mehr anmelden, da er vom Server noch als aktiv registriert ist. Nach diesem Zeitraum würde die Session serverseitig automatisch geschlossen und ein erneutes einloggen wäre möglich.

b) Der Benutzer kann sich sofort wieder anmelden, indem die alte Session durch die neue überschrieben wird. Dabei gehen dann allerdings aktuelle Informationen, wie die Einträge im Forum verloren.

In unserem Projekt wird Variante b) realisiert, da es den meisten Nutzern wichtiger ist, schnellstmöglich bei unabsichtlichen oder fehlerhaften Schließen wieder in das Programm zu kommen. Der Verlust aktueller Daten kann als akzeptabel hingenommen werden, da es sich nur um Kopien von Übungsaufgaben handelt oder weniger wichtige Informationen, wie Nachrichten von anderen Nutzern. Ferner wurde entschieden, dass die Session aller Nutzer, die länger als 30 Minuten inaktiv sind, beendet wird, um sicherzustellen, dass das Programm auch bei einem inkorrekten Abbruch terminiert.

3.4. ...beim Aufbau der Startseite

Bei Gestaltung der Startseite können verschiedene Wege beschritten werden. Im Wesentlichen gibt es zwei Strategien:

a) Es gibt für alle Nutzer die gleiche Startseite, die individuell an die rollenspezifischen Bedürfnisse angepasst ist.

b) Für jeden Nutzer wird eine eigene Startseite entwickelt, die statisch alle gewünschten Menüpunkte und Handlungsmöglichkeiten enthält.

Das Programm folgt dem unter a) beschriebenen Szenario. Dabei wird nach dem Login für jeden Nutzer die gleiche Seite aufgebaut, lediglich das Menü ist unterschiedlich gestaltet. Dafür muss für jeden prinzipiell möglichen Menüpunkt geklärt sein, von welchen Rollen aus er verwendet werden darf. Bei der Erzeugung des Menüs wird dann für jedes Item geprüft, ob es für die Rolle zugelassen ist. Ist dies der Fall, wird die Aktion der Menüleiste hinzugefügt, andernfalls nicht.
Vorteile dieser Variante sind die schnelle und unkomplizierte Anpassung des Menüs im Nachhinein, somit müssen lediglich die Zugriffsrechte neu gesetzt werden, und die einheitliche Gestaltung, die für mehr Übersichtlichkeit sorgt.

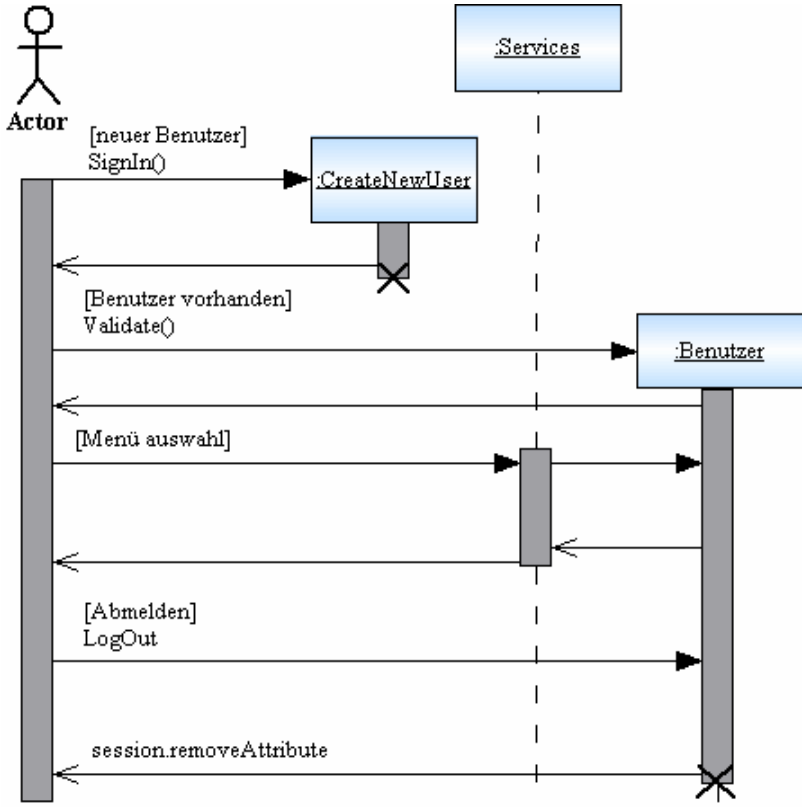

Abb.5: Sequenzdiagramm für den Ablauf einer Benutzersession

3.5. ...bei der Umsetzung der einzelnen Services

Für jeden Nutzer steht im Programm eine Vielzahl von Anwendungen bereit, die jedoch alle einem ähnlichen Aufbau folgen, z.B. sie werden alle durch Servlets realisiert, brauchen grundlegende Daten über den Nutzer und haben die gleichen Attribute.
Dabei stellte sich die Frage in welcher Weise diese Ähnlichkeiten zu Implementieren sind:

a) Es wird das Konzept der Java-Filter ausgenutzt, wobei nach und nach die Anforderungen der einzelnen Servlets analysiert werden.

b) Es wird eine Oberklasse erzeugt, von der alle Services erben und die diese Gemeinsamkeiten beinhaltet.

Eine gemeinsame Oberklasse schien am geeignetsten für das Projekt, da so wichtige Informationen zentral gebündelt sind und schnell für alle Services geändert werden können. Somit wird Alternative b) realisiert.

3.6. ...bei der Gestaltung des Forums

Beim Forum gibt es die Möglichkeit, dieses als Chat oder als Abrufliste zu realisieren. Ein Chat bietet den Vorteil, dass ständig alle Nachrichten sofort gesehen werden. Die Abrufliste hingegen muss aktualisiert werden, um den neuesten Inhalt einzusehen, was den Vorteil bietet, dass der Nutzer nicht ständig belästigt wird, wenn er dies nicht wünscht.

Umgesetzt werden soll eine Abrufliste, in der eine automatische Aktualisierung implementiert werden kann um ein ähnliches Verhalten wie ein Chat zu ermöglichen.

3.7. ...bei der Gestaltung des Menüs

Um eine individuelle Anpassung der Optik an die Wünsche des Betreibers zu gewährleisten werden für die HTML-Seiten CSS verwandt. CSS steht für Cascading Style Sheets.
In einem externen File, dem CSS file, werden bestimmte Stile definiert, die später in den Quellcode der HTML-Seite eingebunden und auf den nachfolgenden Text angewandt werden können. Da diese Datei nicht in den Code des Programms eingebunden ist, kann das Layout jederzeit leicht von „außerhalb“ verändert werden ohne das ganze Programm neu kompilieren zu müssen.
Nachfolgend ist ein kurzes Beispiel für die Verwendung von CSS angegeben:

CSS-File:

```
a {color:red;font-size:13px;}
a:active{color:#444444;}
a:hover {color:yellow;} /* leider nur im IE */

body {padding:0px; SCROLLBAR-FACE-COLOR: #ffffff; SCROLLBAR-HIGHLIGHT-COLOR:
#ffffff; SCROLLBAR-SHADOW-COLOR: #000000; SCROLLBAR-3DLIGHT-COLOR: #006699;
SCROLLBAR-ARROW-COLOR: #006699; SCROLLBAR-TRACK-COLOR: #ffffff; SCROLLBAR-
DARKSHADOW-COLOR: #ffffff; SCROLLBAR-BASE-COLOR: #000000 }

table.menu {width:130px; position:absolute; top:0px; left:0px; background-
color:#ff9900; border:10px solid black;}
table.menu td {background-color:#FFCC00; padding:5px;}

table.border{border:1px solid black;}
table.border td{border:1px solid black; padding-left:5px; padding-
right:5px;margin:0px;}
```

```
div.r{position:absolute; top:0px; left:140px; margin:0px;}
div.r{background-color:#FFCC00; padding:5px;border:10px solid black;}
div.tab{position:relative; padding-left:15px;}
.error{color:red;}
```

Beispiel Quellcode einer HTML-Seite:

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=windows-1252">
    <TITLE>Willkommen</TITLE>
    <link rel="stylesheet" type="text/css" href="theme/style.css">
  </HEAD>
  <BODY LANG="de-DE">
<table class="menu">
  <tr>
    <td>1234567</td>
  </tr>
  <tr>
    <td>
      <a href="Welcome">Willkommen</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="ViewPoints">Punkteübersicht</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="ChangeData">Daten &auml;ndern</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="Logout">Logout</a>
    </td>
  </tr>
</table>

<div class="r">
</div></body>
</html>
```

HTML-Seite:

8965960

Willkommen

Punkteübersicht

Punktestand

Daten ändern

Logout

Text

Abb. 6: HTML-Seite mit CSS-Layout

4. Paket- und Klassenstruktur

Das Projekt ist in vier Packages gegliedert:

- **main:** Im Package main sind alle Klassen enthalten, die für die Einwahl in den internen Bereich, sowie für die Anmeldung an das Projekt nötig sind
- **services:** Alle Klassen, die für die Ausführung von Aktivitäten verantwortlich sind werden im Package Services zusammengefasst
- **util:** Packet in dem verschiedene Hilfsklassen vereint sind.
- **dtype:** hier sind Klassen enthalten, die die verschiedenen Datentypen festlegen

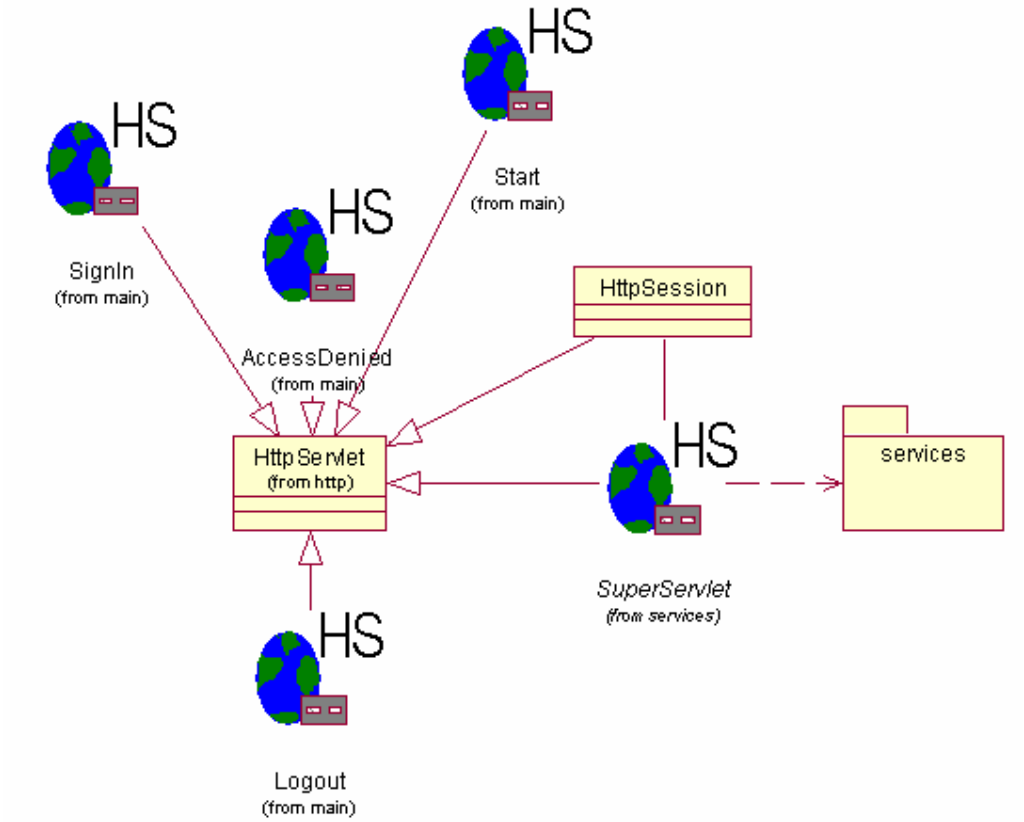

Abb. 7: Klassendiagramm des Projekts Übungsbetrieb im allgemeinen Aufbau

4.1. main

Im Packet main sind die Klassen enthalten, die für den Login oder die Neuanmeldung an das Projekt benötigt werden. Das Servlet Start überprüft die Benutzereingaben die in das Login Formular eingegeben wurden und vergleicht diese mit den Werten in der zum Benutzer gehöriger user.xml. Ist der Benutzer nicht vorhanden oder sind die Logindaten nicht richtig erhält er eine Fehlermeldung. Der Benutzer kann auf der Startseite wenn er noch nicht registriert ist, das Servlet SignIn aufrufen.

4.2. Services

Den Grossteil des Programms macht das Package Services aus. In ihm sind alle Klassen zusammengefasst, die zur Ausführung programmeigener Aktivitäten benötigt werde. Dazu gibt es eine Klasse SuperService, die die Grundfunktionalitäten aller Services beinhaltet und von der alle speziellen Services erben:

- GetServices: Beinhaltet alle Operationen zum Auslesen von Daten
  - GetArchive: Archivdatei erstellen
  - GetConfig: Config Datei auslesen
  - GetDownloadData: Informationen über die vorliegenden Dokumente auslesen
  - GetExamData: Klausurdaten auslesen
  - GetForum: Nachrichten aus der forum.xml auslesen
  - GetGroupData: Daten für die Übungen aus der veranstalltungen.xml auslesen
  - GetPublic: liest die Daten für den öffentlichen Bereich aus
  - GetSeriesCorrection: Punkte aus XML Datei auslesen
  - GetSeriesData: Daten für die Aufgaben auslesen (z.B. Abgabetermin)
  - GetService: Benutzer Daten aus user.xml auslesen
  - GetSolutionList: erstellt Liste mit Übersicht über die abgegebenen Lösungen
  - GetUploadService: Diese Klasse stellt Assistenzmethoden für den Upload bereit
  - GetUserData: Userdaten auslesen aus der jeweiligen user.xml
  - GetWelcomeSite: Text für die Willkommens Seite wird ausgelesen
  - MultipartRequest: Empfängt requeststream und erstellt daraus die zu erstellende Datei

- SetServices: dient dem Schreiben und Speichern von Daten und vererbt an folgende Klassen:
  - CreateNewUser: erstellt Dateien/Pfade nach erfolgreichem SignIn
  - SetArchive: Archivdatei wiederherstellen und an das Projekt übertragen
  - SetConfig: Mail Config Datei schreiben
  - SetExamData: Funktionen zur Klausuranmeldung
  - SetForum: Nachrichten in die forum.xml schreiben
  - SetGroupData: Daten für die Übungen in die veranstalltungen.xml schreiben
  - SetKorrekturdaten: schreibt eingegebenen Korrekturdaten in die korrekturdaten.xml
  - SetPublic: Daten für den externen Bereich schreiben
  - SetService: stellt Assistenzmethoden für das schreiben von XML Dateien bereit
  - SetSolutionList: bei Abgabe einer Lösung Eintrag in die Abgabeliste
  - SetUploadService: Methoden zum Upload von Dokumenten
  - SetUserData: Userdaten in der jeweiligen user.xml speichern
  - SetWelcomeSite: Daten für die Willkommenseite in die uebung.xml schreiben

- filetransfer: Hier werden alle Funktionen zum Upload von Dateien zur Verfügung gestellt.
  - UploadTest: Datei Hochladen und speichern

- mail: Funktionen die zum versenden von e-Mails nötig sind
  - SendMail: e-Mail an eine spezifische Adresse senden
  - SMTPAuthenticator: benötigte Klasse zum Mail Versand

- AddUser: Klasse zum hinzufügen eines Benutzers
- AllPoints: erstellt die Klarliste für den Dozenten
- Backup: Funktionen zum wiederherstellen des Projektes mittels Backup Dateien
- ChangeData: Ändern der persönlichen Daten
- CheckInExam: ermöglicht anmelden an Klausuren
- Configure: e-Mail Daten bearbeiten
- Deinstallation: Funktionen um Deinstallation zu verwirklichen

- DeleteUser: Benutzer löschen
- DownloadStudent: Funktionen für den download von Dateien, die per Fileupload hochgeladen wurden
- EditExam: Klausurdaten verändern bzw. erstellen
- EnterPoints: Klasse die das Eintragen der Punkte zur Verfügung stellt
- FileUpload: Dateien auf den Server übertragen
- Forum: Klasse für Funktionen die für die Kommunikation innerhalb des Forums nötig sind
- ManageGroups: Übungsgruppenverwaltung wird hier implementiert
- ManagePublic: bearbeiten der Startseite des öffentlichen Bereiches wird hier implementiert
- ManageWelcome: bearbeiten der Welcome Seite wird hier implementiert
- RegistrationSeminar: Übungsgruppen wählen, wechseln
- Service: Userdatei laden bzw. testen ob vorhanden
- SolutionDownload: Methode um mit abgegebenen Lösungen weiter zu verfahren
- UploadSolution: stellt Funktionen zum Upload der Lösungen durch den Studenten bereit
- ViewPoints: Punkte für den jeweiligen Studenten auslesen
- Welcome: Startseite im internen Bereich des Programms

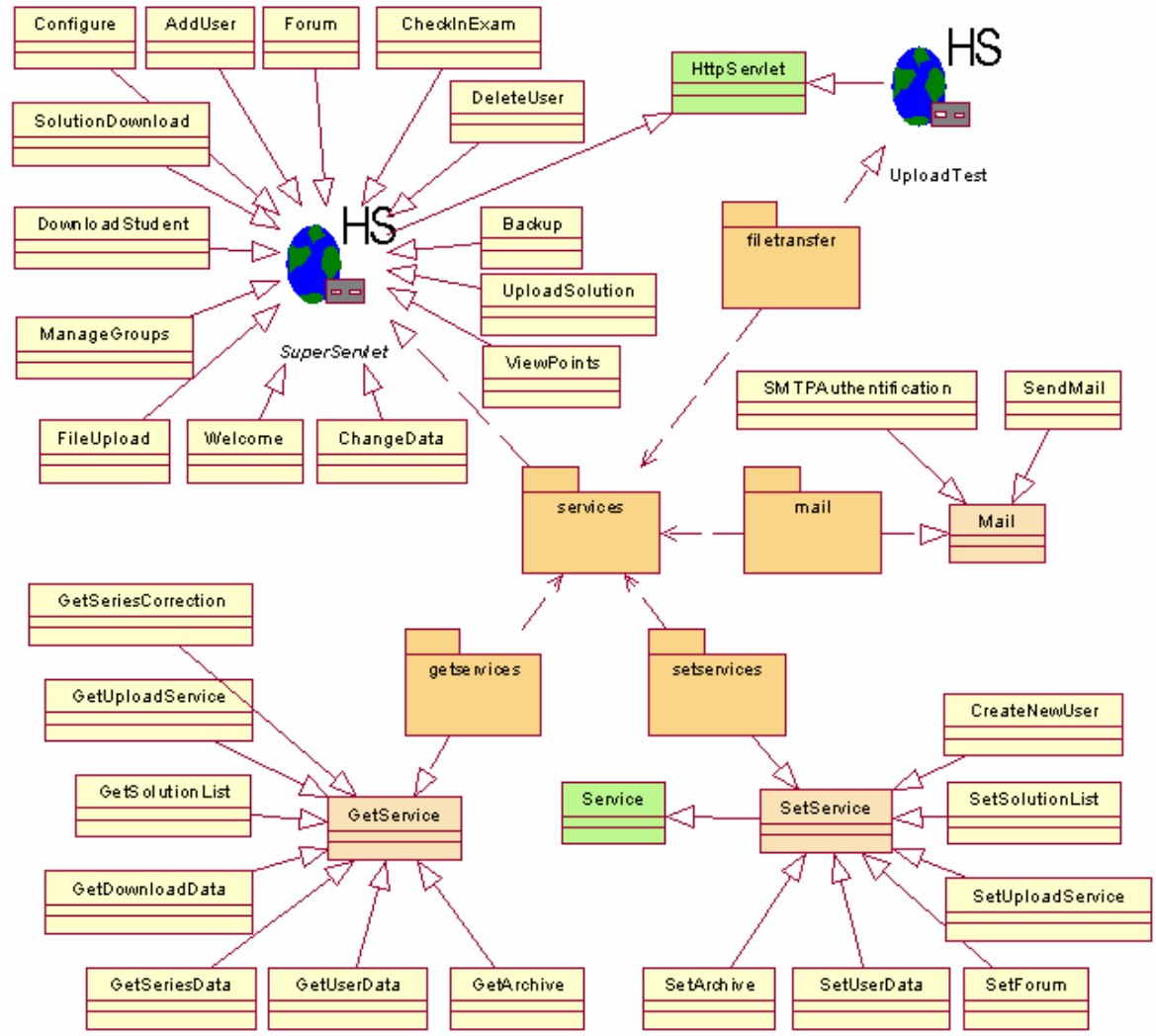

Abb. 8: Übersicht über die Klassenstruktur von Services- aus Platzgründen wurden nicht alle Servlets und Klassen aufgeführt. Die Übersicht darüber befindet sich über der Abbildung.

Als Beispiel für die Services zeigen wir ChangeData und UploadSolution als Klassendiagramm:

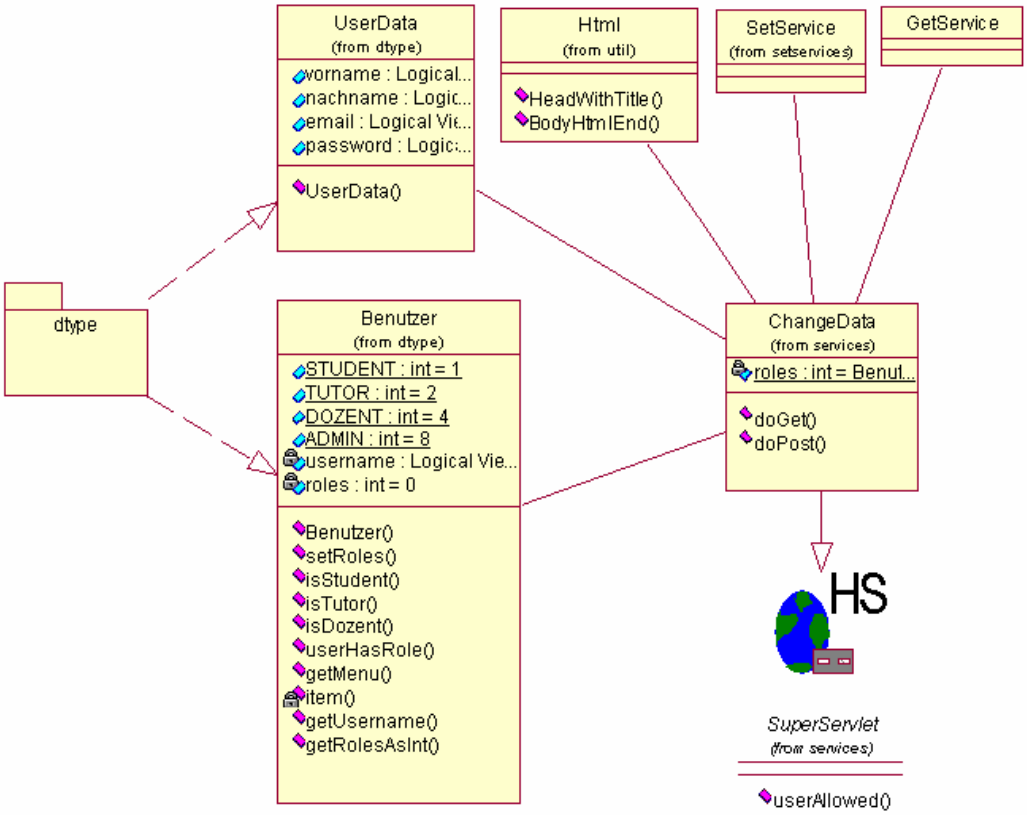

Abb 8: Klassendiagramm ChangeData

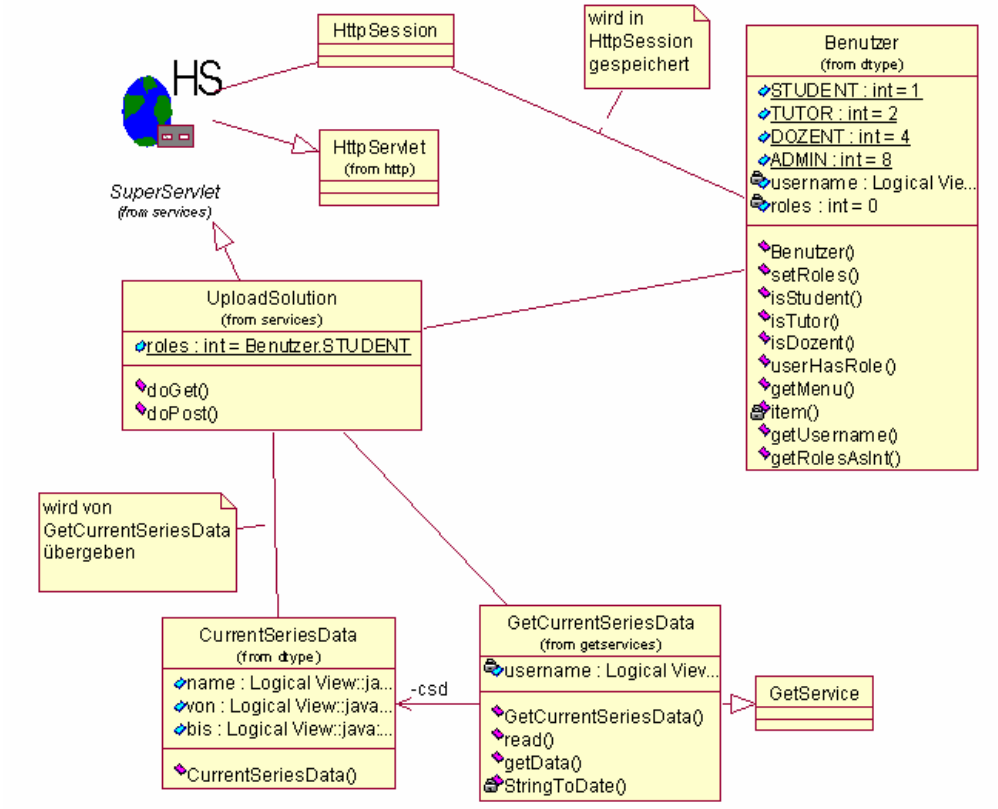

Abb 9: Klassendiagramm UploadSolution

4.3. util

Ständig wiederkehrende Funktionen werden als Assistenzklassen implementiert. Diese werde hier zusammengeführt. Dazu gehören Klassen die zum Beispiel den Head und den Tail einer HTML Datei schreiben oder das Arbeitsverzeichnis des Servers ausgeben.

4.4 dtype

Für das Projekt werden verschiedene Datentypen benötigt, die extra definiert werden. Der Übersicht halber werden dies hier in einem Packet zusammengefasst.

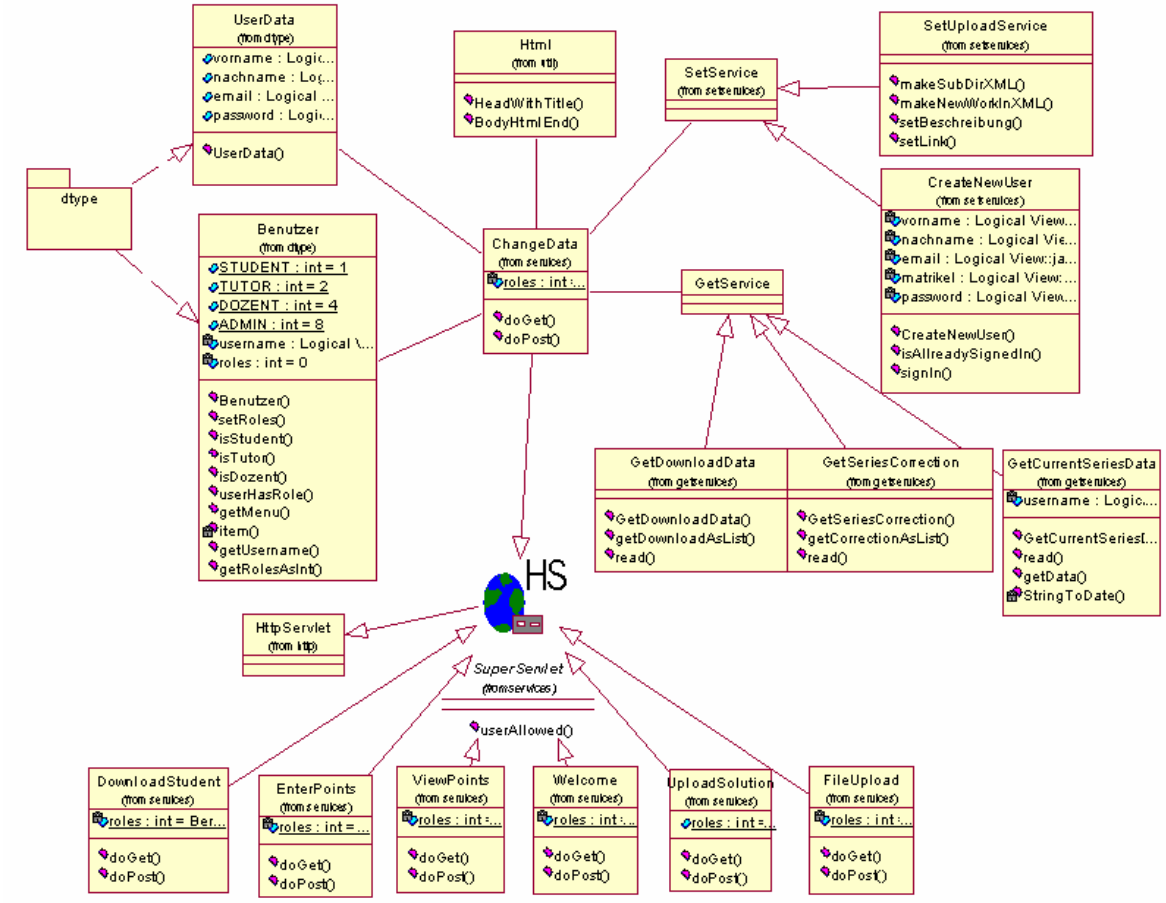

Abb. 10: Klassendiagramm dtype mit Darstellung einiger weiteren Ebenen der Klassenstruktur

5. Pfadstruktur:

Sämtliche zu speichernden Daten oder erhältlichen Informationen sind im Verzeichnis data gespeichert in jeweils verschiedenen Unterordnern.

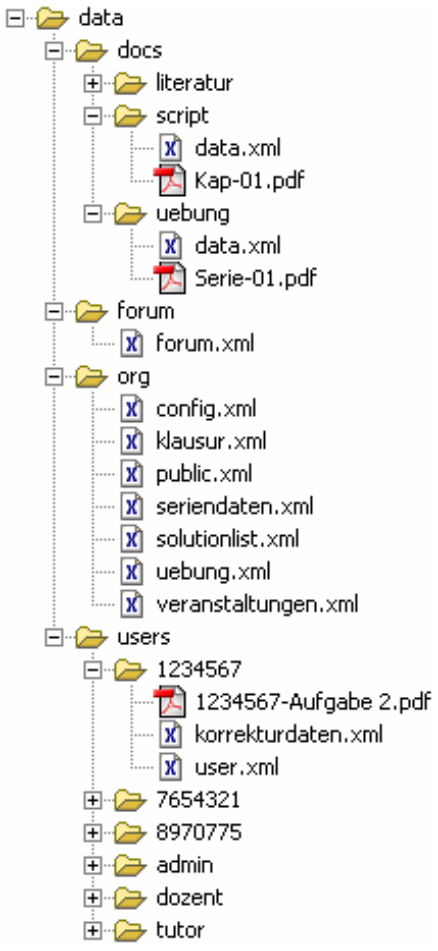

Abb. 11: Pfadstruktur

Unter docs gibt es Standardmäßig die Ordner literatur, script und uebung in denen die jeweiligen Dateien für den Übungsbetrieb gespeichert werden. Es können aber auch noch zusätzliche Ordner erstellt werden, wenn diese benötigt werden. Außerdem existiert in jedem Ordner eine data.xml in der alle vorhandenen Dateien eingetragen sind.

Im Ordner Forum werden alle Forumeinträge in der Forum.xml gespeichert

Der Ordner org enthält immer diese Dateien, die alle wichtigen zu speichernden Informationen zum Übungsbetrieb enthalten.

Unter users werden Unterordner angelegt, die mit dem jeweiligen Loginnamen des Benutzers übereinstimmen. Dort wird immer einer user.xml angelegt, in der alle persönlichen Informationen des Benutzers gespeichert sind. Korrekturdaten.xml beinhaltet die Korrekturergebnisse für alle Serien. Außerdem werden die zu dem jeweiligen Studenten gehörigen Lösungen in diesem Ordner nach der festgelegten Schablone („Username“-„Übungsserie“.pdf) gespeichert.