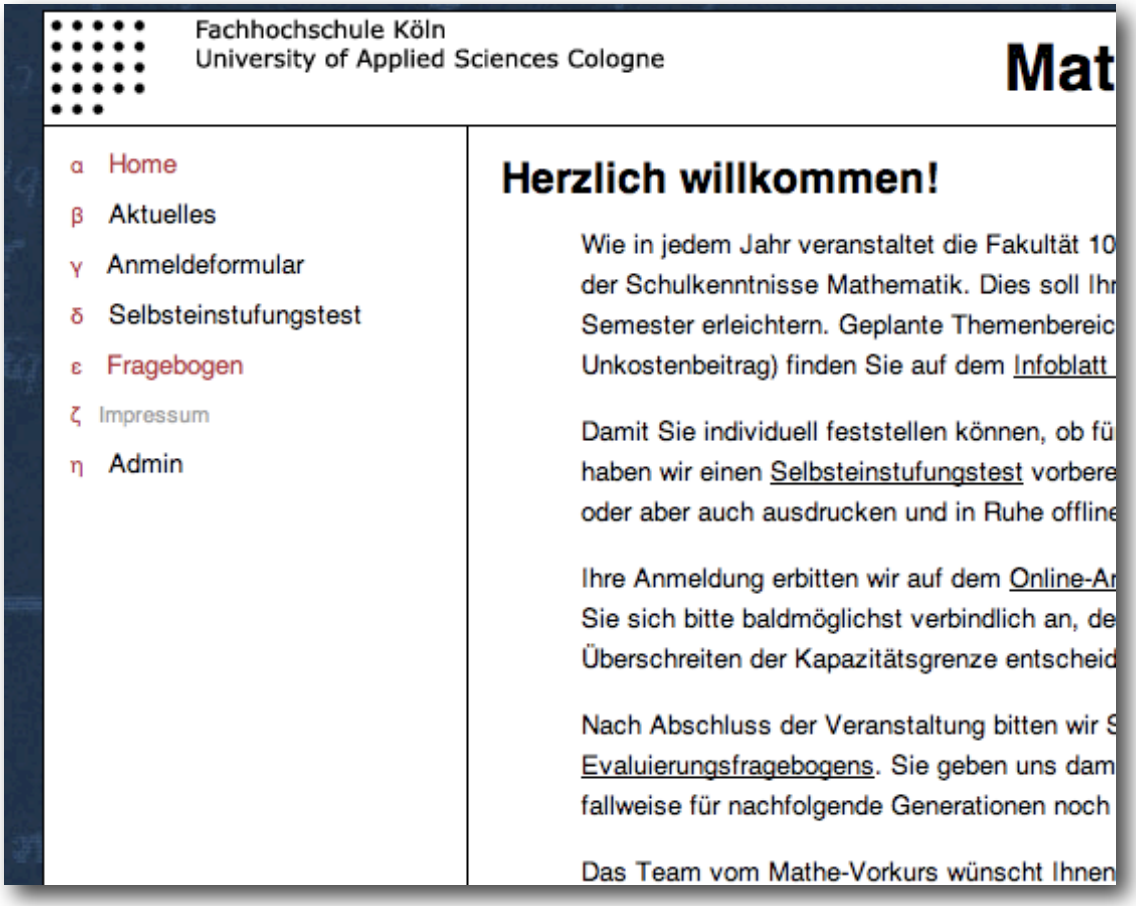
QQ2 Projekt: Online Anmelde-tool für den Mathematikvorkurs der Fachochschule Köln-Campus Gummersbach

Erstellt von: Norma Yvonne Sappelt • Kus, Jan
Bertreut von: Wolfgang Konen



Index

Die Idee

Das Projekt, die Zielgruppe und Ziel

Das entstandene QQ2 Projekt, Entwicklung eines Web-Portals mit Anmeldetool und Einstufungstest für den Mathematik-Vorkurs, ist Ergebnis des Optimierungsprozesses der Formalitäten rund um den Mathematik- Vorkurs. Das Ziel welches wir dabei verfolgten, war es diese durch den schnellen und unkomplizierten Zugang zum Internet zu verringern.

Die dynamisch gestaltete Internetseite demonstriert einen Auftritt welcher eine Orientierungsfunktion übernimmt.

Sie setzt sich aus einen Selbsteinstufungstest, einem Anmeldeformular und einem Fragebogen zusammen.

Hierbei wird eine bestimmte Zielgruppe von Benutzern, den Bewerben angesprochen, Es sind Benutzer die die Chance und die Möglichkeit nutzen möchten, welche das Web-Portal ihnen bietet, ihre Mathevorkenntnisse einzustufen und sich gegebenenfalls an einem Mathematik-Vorkurs anzumelden, gleichzeitig haben wir das Ziel mittels des Fragebogens eine Verbesserung der Veranstaltung zu ermöglichen.

Die Realisation dieser Applikation fand auf Basis von Standardtechnologien, wie (X)HTML/CSS, Java Server Pages, JavaScript und einer ORACLE Datenbank statt.

Der Fokus dieses Projektes wurde unsererseits auf die Möglichkeiten zur frühzeitigen Orientierung neuer Studenten im Bereich Mathematik gerichtet.

Distribution

Da das Anmeldeportal für Zwecke der Fachhochschule Köln genutzt werden soll, ist es nicht vorgesehen das Portal in irgendeiner Weise zu verbreiten. Es soll ausschliesslich über das Internet, auf den Servern der Fachhochschule Köln genutzt werden.

Umsetzung

Eingesetzte Hard- und Software

Zur Erstellung des Projektes wurde folgende Software verwendet:

- Gimp- zur Erstellung von Graphiken für das Design
- Texrwangler- für das Skripten von JSP und SQL Code
- iSqlPlus- zum erstellen von Tabellen in der Datenbank

Gearbeitet wurde hauptsächlich auf privatem Equipment. Dieses bestand aus;

- PC Notebook
- iBook G4

Welche Systeme unterstützt die Fachhochschule in solchen Projekten?

Die Fachhochschule Köln (Campus Gummersbach) verfügt über verschiedene Implementierungs und Laufzeitsysteme in denen man eine solche Anwendung entwickeln kann. Es wird aber aus Sicherheitsgründen folgende Konfiguration für ein Web-System wie das Matheportal empfohlen:

- TOMCAT-Webserver (Unterstützung von JavaServer Pages)
- ORACLE-Datenbank

Mehr braucht dieses System nicht. Der TOMCAT-Webserver ist ein Derivat des Apache-Webservers der kommerziell auf der ganzen Welt eingesetzt werd. Daher interpretiert der TOMCAT-Server sowohl .jsp als auch .html und .css Dateien.

Die Webseite, das HTML, CSS und JavaScript

Die einzelnen Seiten des Mathe-Vorkurs-Portals sind in HTML geschrieben, diese HTML-Seiten sind mit CSS formatiert. Auf den HTML-Code werden wir hier nicht näher eingehen, da dieser ja selbsterklärend ist.

Für die Layoutformatierung haben wir uns für die Cascading Stylesheets entschieden, da uns so die Möglichkeit gegeben ist, das Format der Seite zentral zu editieren, was unnötige Arbeit vermeidet.

In den Selbsteinstufungstest wurde Java Skript eingebunden, mit dem die direkte Auswertung des Testes möglich ist.

Der Ablauf:

Sobald der User auf der Seite des Selbsteinstufungstests (alle) Fragen beantwortet hat und den Submit-Button betätigt, tritt das angesprochene Java Skript in Aktion.

Zuerst sind alle Antworten im JavaScript automatisch auf „null“ gesetzt und mittels der Eingabe des Users und Absenden des Formulars werden auf der folgenden Seite Antworten aus der URL des „GET-Requestes“ extrahiert.

Im Anschluss daran werden die Antworten des Benutzers mit den richtigen Lösungen verglichen (im Java Script) und die Übereinstimmungen gezählt.

Aufgrund dieser Daten wird nachfolgend die Ausgabe generiert.

Die Datenbank: Überblick über die Tabellen

Die Datenbank der Fachhochschule basiert auf einem ORACLE Datenbank Management System. Als Frontend für das erstellen der Tabellen diente iSQLPlus, das auf den Servern der Fachhochschule verfügbar ist (http://oras1.gm.fh-koeln.de:7777/isqlplus).

Die Realisierung verfolgte das Ziel, Daten vom Inhalt zu trennen. Im folgendem werden die, in der Datenbank vorhandenen Tabellen in ihren SQL ‘CREATE TABLE’ Befehlen aufgelistet und kurz beschrieben:

```
CREATE TABLE  anmeldeliste (
 anmeld_id NUMBER NOT NULL PRIMARY KEY ,
 anmeld_nachname VARCHAR( 30 ) NOT NULL ,
 anmeld_vorname VARCHAR( 30 ) NOT NULL ,
 anmeld_geb date NOT NULL ,
 anmeld_matnr varchar( 50 ) ,
 anmeld_stud_kuerzel VARCHAR( 50 ) NOT NULL ,
 anmeld_strasse VARCHAR( 75 ) NOT NULL ,
```

```
anmeld_land varchar( 30 ) NOT NULL ,
 anmeld_plz varchar( 20 ) NOT NULL ,
 anmeld_ort VARCHAR( 75 ) NOT NULL ,
 anmeld_tel VARCHAR( 30 ),
 anmeld_email VARCHAR( 75 ) NOT NULL ,
 anmeld_jahr date
)
```

Die Tabelle ‘anmeldeliste’ enthält alle Daten der abgesicherten Teilnehmer. Wichtig hierbei ist das Feld `anmeld_jahr date` zu beachten, dass für jeden Teilnehmer Informationen über das Anmeldejahr enthält, wodurch eine einfache Ausgabe der im jeweiligen Jahr angemeldeten Teilnehmer unproblematisch möglich ist.


```
create sequence anmeldeliste_seq
start with 1
increment by 1
nomaxvalue;
```

Die Tabelle ‘anmeldeliste’ wird durchgehend von einer Sequenz `anmeldeliste_seq` ‘begleitet’, die zuständig dafür ist, jedem angemeldetem Teilnehmer eine eindeutige ID zuzuordnen. Nach jeder Anmeldung wird diese Sequenz um eins inkrementiert, was das fatale entstehen von doppelten ID’s verhindert.

Ein Beispiel eines INSERTS für einen Teilnehmer;

```
insert into anmeldeliste
values(anmeldeliste_seq.nextval,'koos','jan','22-12-1999','11042
239', 'mi', 'vierkotterfeld 7','51503',
'roesrath','germany','0049-2205-911001','koos@jbv-koos.de', sys-
date);
```

Für die Verwaltung der Anmeldungen, ist ein entsprechendes Login mit einem dazugehörendem Password nötig. Diese Daten sind (unverschlüsselt) in der Tabelle ‘administratoren’ abgelegt:

```
create table administratoren (
admin_id number(1) not null primary key,
admin_login varchar(30) not null,
admin_passwd varchar (30) not null,
admin_name varchar(30) not null,
```

```
admin_nachname varchar (30) not null);
```

Ein weiterer Administrator kann mit folgender Anweisung hinzugefügt werden;

```
insert into administratoren values (1, 'LOGIN', 'PASSWORD','NA-
ME', 'NACHNAME');
```

Ein weiteres Login ist für das Absenden eines Fragebogens nötig. Diese Daten werden wiederrum in der 'qq2_user' gespeichert:

```
create table qq2_user (
qq2_user_id number(1) not null primary key,
qq2_user_login varchar(30) not null,
qq2_user_passwd varchar (30) not null,
qq2_user_name varchar(30) not null,
qq2_user_nachname varchar (30) not null);
```

Um den Zugriff auf die Anmeldemöglichkeit einzuschränken zu können, wurde eine spezielle Tabelle angelegt, die die Anmeldemöglichkeit verhindern (bei z.B. Überfüllung) oder auch freischalten kann. Diese Tabelle wird vom Administrator verwaltet.

```
CREATE TABLE  block_table (
 blockt_flag VARCHAR( 1 ) NOT NULL ,
 blockt_text VARCHAR( 75 ) NOT NULL ,
 blockt_date date NOT NULL,
 blockt_user varchar( 30 ) not null,
);
```

Nun folgen eine Reihe von Tabellen, die das Absichern und Evaluieren des Fragebogens dienen.

Als erstes soll hier die Tabelle 'fragebogen_fragen' und ihre INSERT Befehle aufgeführt werden. In der aktuell sich online befindenden Version (v. 9.0.1) werden diese Informationen nicht aktiv verwendet.

```
CREATE TABLE  fragebogen_fragen (
 frage_id NUMBER(2) NOT NULL PRIMARY KEY,
 frage_text VARCHAR( 100 ) NOT NULL
 );
```

```
insert into fragebogen_fragen values(1, 'Studiengang');

insert into fragebogen_fragen values(2, 'Geschlecht');

insert into fragebogen_fragen values(3, 'Alter');

insert into fragebogen_fragen values(4, 'Wie war der Umfang der
Vorlesung?');

insert into fragebogen_fragen values(5, 'Wie war der Inhalt der
Vorlesung?');

insert into fragebogen_fragen values(6, 'Wie war die Präsentati-
on der Vorlesung?');

insert into fragebogen_fragen values(7, 'Es wurden genug Themen-
gebiete angesprochen?');

insert into fragebogen_fragen values(8, 'Wie war der geforderte
Lern- und Arbeitsaufwand?');

insert into fragebogen_fragen values(9, 'Wie waren die Litera-
turhinweise?');

insert into fragebogen_fragen values(10, 'Wie war der Umfang der
Übungsaufgaben?');

insert into fragebogen_fragen values(11, 'Wie war der Inhalt der
Übungsaufgaben?');

insert into fragebogen_fragen values(12, 'Wurden die Lösungen
verständlich vorgetragen?');

insert into fragebogen_fragen values(13, 'Wurde genug Raum für
Diskussionen gelassen?');

insert into fragebogen_fragen values(14, '...gingen auf Einwän-
de/Fragen gut ein');

insert into fragebogen_fragen values(15, '...machten einen moti-
vierten Eindruck');

insert into fragebogen_fragen values(16, '...waren gut vorberei-
tet');

insert into fragebogen_fragen values(17, 'Hat der Vorkurs etwas
gebracht?');

insert into fragebogen_fragen values(18, 'Haben Sie im Vorfeld
Ihre Mathematikkenntnisse...');

insert into fragebogen_fragen values(19, 'Was für Themengebiete
haben Ihnen noch gefehlt?');
```

```
insert into fragebogen_fragen values(20, 'Für eigene Kommentare
steht Ihnen noch dieses Textfeld zur Verfügung');
```

Das Einfügen von den abgesendeten online-Umfragebögen wird durch eine Reihe von einzelnen INSERT Befehlen in verschiedenen Tabellen abgesichert, die nun aufgeführt werden.

Als erstes wird mit jedem Eintrag die Tabelle 'fragebogen_person' mir den jeweiligen Daten gefüllt und eine Sequenz weist diesem Eintrag eine eindeutige ID zu (analog wie bei der Anmeldung). Diese ID wird bei weiteren INSERT Befehlen des jeweiligen Eintrags als Bezugsvariable herangezogen, damit auch jeder abgesendeter Fragebogen von allen anderen unterscheidbar ist- was natürlich wichtig für die gewollte Statistik ist. Die Tabelle enthält (anonyme) Daten der Person, die den Fragebogen gerade ausgefüllt und abgesendet hat.

```
CREATE TABLE  fragebogen_person (
 person_id NUMBER NOT NULL PRIMARY KEY ,
 person_stud_kuerzel VARCHAR( 10 ) NOT NULL ,
 person_geschlecht      VARCHAR( 1 ) NOT NULL,
 person_alter NUMBER( 2 ) NOT NULL
 );

create sequence fragebogen_person_seq
start with 1
increment by 1
nomaxvalue;
```

Die folgenden drei Tabellen (`fragebogen_antworten, fragebogen_gewuenschte_themen, fragebogen_anregungen`) wiederrum speichern alle Antworten, fehlende Themen und Anregungen der jeweiligen Person ab. In der Tabelle wird schliesslich das jeweilige Datum abgesichert, um die Daten von verschiedenen Jahrgängen, wie auch schon bei der Anmeldung gemacht wurde, auseinander zu halten.

```
 CREATE TABLE  fragebogen_antworten (
 frage_id NUMBER(2) NOT NULL,
 person_id NUMBER NOT NULL,
 antwort NUMBER(1) NOT NULL,
 FOREIGN KEY (person_id) REFERENCES fragebogen_person(per-
son_id),
 PRIMARY KEY (frage_id, person_id)
 );

 CREATE TABLE  fragebogen_gewuenschte_themen (
```

```
 person_id NUMBER NOT NULL,
 thema VARCHAR(50),
 FOREIGN KEY (person_id) REFERENCES fragebogen_person(per-
son_id));

CREATE TABLE fragebogen_anregungen (
 person_id NUMBER NOT NULL,
 anregung_text VARCHAR(4000),
 FOREIGN KEY (person_id) REFERENCES fragebogen_person(per-
son_id));

CREATE TABLE fragebogen_datum (
 person_id NUMBER NOT NULL,
 datum DATE,
 FOREIGN KEY (person_id) REFERENCES fragebogen_person(per-
son_id),

 PRIMARY KEY (datum, person_id));
```

Die Auswertung der abgesicherten Daten erfolgt für jeden Studiengang einzeln von Frage 1 - 18.

Als erstes werden mir folgendem SELECT Befehl die Geschlechter ermittelt (mithilfe einer Oracle spezifischen 'case' Anweisung. Diese Ergebnisse werden mittels 'count' gezählt und können so auf verschiedene Weise weiterverarbeitet werden (z.B. JSP). Hier ein Beispiel für das zählen der männlichen und weiblichen Teilnehmer der Medieninformatik:

```
select count(case when person_geschlecht = 'm' then 1 else null
end ) as geschlecht_maennlich,

count(case when person_geschlecht = 'w' then 1 else null end )
as geschlecht_weiblich,

avg(person_alter) from fragebogen_person where person_stud_kuer-
zel = 'mi';
```

Anschliessend erfolgt das Evaluieren von den restlichen Antworten (4-18). Hier werden die jeweiligen Antwortmöglichkeiten auch mittels der Oracle spezifischen 'case' Anweisung gewonnen und gezählt. Damit aber auch wirklich jede Person auch nur einmal gezählt wird, werden mittels einer 'where' Bedingung alle (in diesem Beispiel) 'mi' Studenten selektiert und mir der 'fragebogen_antworten' Tabelle verknüpft.

```
select fa.frage_id, count(case when fa.antwort = '1' then 1 else
null end ) as antwort_gut, count(case when fa.antwort = '2' then
1 else null end ) as antwort_weissnicht, count(case when
fa.antwort = '3' then 1 else null end ) as antwort_schlecht from
```

```
fragebogen_antworten fa, fragebogen_person per where fa.frage_id
IN (4,5,6,7,8,9,10,11,12,13,14,15,16,17,18) and fa.person_id =
(select per.person_id from fragebogen_person where
per.person_stud_kuerzel = 'mi' group by per.person_stud_kuerzel)
group by frage_id order by fa.frage_id asc;
```

Zum Schluss noch 2 Abfragen über alle Daten der Fragebogentabelle um die durch die Studenten gemachten Angaben über die gewünschten Themen und Anregungen zu erhalten.

```
select thema from fragebogen_gewuenschte_themen group by thema;
```

```
select anregung_text from fragebogen_anregungen group by anre-
gung_text;
```

Verwendung von Java Server Pages (JSP)

Diese Anwendung entstand unter anderem unter Verwendung von Java Server Pages. JSP ist eine Skriptsprache für die Entwicklung von Webanwendungen (entstanden 1999) die es erlaubt unteranderem Datenbanken abzufragen, und für die Zwecke wurde JSP auch hier verwendet. JSP basierte Anwendungen werden über einen 'Tomcat' Webserver ausgeführt. Auf der Fachhochschule läuft so ein Webserver unter der Adresse:
http://ora51.gm.fh-koeln.de:8000/index.jsp

Mit Java Server Pages ist das MVC Paradigma sehr gut realisierbar, was wir auch hier versuchten anzuwenden. Hierum geht es die Funktionalität (Controll) von dem Design (View) zu trennen.

Im allgemeinem ist unser ganzes (ausgenommen der Selbsttest) Tool in JSP geskriptet worden. Es besteht aus 30 .jsp Dateien die sich imaginär in drei Gruppen aufteilen lassen:

- Administrator JSP-Seiten
- Fragebogen JSP-Seiten
- Anmelde JSP-Seiten

Hier soll zunächst nur grob die Funktionsweise von JSP an einem Beispiel beschrieben werden, Schauen wir uns ein Stück Code der Datei 'anmeldung.jsp' an, die in unserer Anwendung als ein typischer 'View' interpretiert werden kann, da sie (fast) nur die Sicht auf die Anmeldung für den User darstellt:

```
<%@ page language="java" contentType="text/html" %>
```

```
<%@ page import="java.util.*" %>

<%@ taglib prefix = "c" uri="http://java.sun.com/jstl/core" %>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt" %>

<%@ taglib prefix="ora" uri="orataglib" %>

<%@ taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>

<sql:query var="metaData">

  SELECT * FROM block_table

</sql:query>

<c:choose>

<c:when test="${metaData.rows[0].blockt_flag == 0}">

  <c:redirect url="index_block.jsp">

    <c:param name="block_text"
value="${metaData.rows[0].blockt_text}" />

  </c:redirect>

</c:when>

</c:choose>

     (...)

<c:if test="${nachnameError != 'invalid'}"><input type="text"
size="30"name="nachname" value="<c:out value="${param.nachname}"
/>"></c:if>
```

Als allererstes fallen die sogenannten ‘Taglibs’ auf:

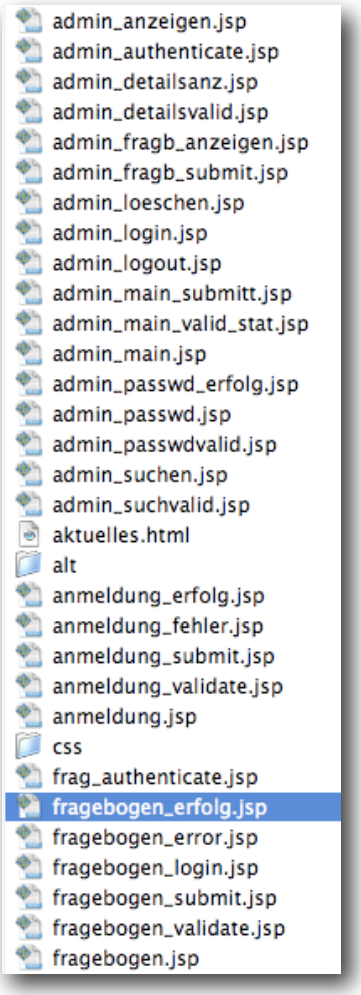
```
<%@ taglib prefix = "c" uri="http://java.sun.com/jstl/core" %>
```

Dies sind spezielle JSP Bibliotheken, die man verwenden kann um bestimmte Funktionen (wie Schleifen, Anweisungen etc.) verwenden kann. Die zwei wichtigsten Taglibs in unserer Anwendung waren das ‘core’ und ‘sql’ Taglib. Das ‘core’ ermöglicht das ausführen von if-Abfragen, z.B.:

```
<c:if test="${nachnameError == 'invalid'}">

<input type="text" size="30"name="nachname" value="<c:out
value="${param.nachname}" />"><br><span class="errnotes">Der
Nachname muss angegeben werden (max. 30 Zeichen)</span>

</c:if>
```

oder auch einfache choose- und when-Anwweisungen:

```
<c:choose>

<c:when test="${metaData.rows[0].blockt_flag == 0}">

  <c:redirect url="index_block.jsp">

    <c:param name="block_text"
value="${metaData.rows[0].blockt_text}" />

  </c:redirect>

</c:when>

</c:choose>
```



Das ‘sql’ Taglib sorgt für Datenbankabfragen jeglicher Art.

Hier im Beispielcode von der Datei ‘anmeldung.jsp’ wird am Anfang über eine einfache SQL Abfrage die Tabelle ‘Block_Table’ abgefragt, um festzustellen ob die Anmeldeliste offen ist oder durch den Administrator gesperrt wurde. Die Anfrage wird mittels des <sql:query> Tags geöffnet und mit </sql:query> wieder geschlossen:

```
<sql:query var="metaData">

  SELECT * FROM block_table

</sql:query>
```

Zwischen den Tags können jeweils nur eine Abfrage stattfinden. Möchte man eine Reihe von Abfragen starten (oder INSERT’s), ist eine Verwendung des <sql:transaction> sinnvoll.

Wie man schnell bemerken kann, wird hier JSP mit HTML vermischt. Dies ist auch möglich, und bei ‘Views’ unvermeidlich.

Nachdem der Benutzer die Eingaben in der anmeldung.jsp getätigt hat, werden die Eingabeparameter der ‘anmeldung_validate.jsp’ übermittelt. Diese prüft diese Eingaben auf Richtigkeit, damit es später keine Exceptions beim füllen der Datenbank gibt. Unteranderem wird hier auch mit regulären Ausdrücken gehandelt, da JSP auch das einbetten von Java Code ermöglicht. Dies hat den Vorteil dass auch bequem auf die korrekte Angabe von Adressen und Telefonnummern, als auch Matrikelnummern etc. geprüft werden kann. Jedes Eingabefeld wird einzeln geprüft und je nach Ergebnis eine Variable auf ‘true’ oder ‘false’ gesetzt. Wenn irgendeine Eingabe fehlerhaft war, wird der User zur ‘anmeldung.jsp’ zurückgeführt. Die

Überprüfung der E-Mail Adressen wird hier noch zusätzlich das ‘orataglib’ verwendet. Um reguläre Ausdrücke und Java mit JSP zu benutzen zu können werden noch folgende Java Bibliotheken benötigt:

```
<%@ page import="java.util.*" %>
<%@ page import="java.util.regex.*" %>
```

Ein Beispiel für die Validierung von Benutzereingaben sähe so aus:

```
 <%Pattern stadt_p = Pattern.compile("^.{1,75}$");

   Matcher stadt_m =
stadt_p.matcher(request.getParameter("stadt"));%>

<c_rt:if test='<%=!stadt_m.matches()%>'>

  <c:set var="stadtError" scope="request" value="invalid" />

  <c:set var="isValid" value="false" />

</c_rt:if>


<ora:ifValidEmailAddr value="${param.mail}"

  var="isValidEmail" />

  <c:if test="${!isValidEmail}">

    <c:set var="mailError" scope="request" value="invalid" />

    <c:set var="isValid" value="false" />

  </c:if>
```

Nachdem eine erfolgreiche Prüfung der Eingaben vollendet ist, werden die Daten an die nächste JSP Datei übermittelt, die sich mit dem Einfügen der Daten in die Datenbank beschäftigt. In der ‘anmeldung_submit.jsp’ wird als erstes geprüft (mittels einer SELECT Abfgare im <sql:query> Tag) ob ein User schon vorhanden ist, oder nicht, die gesuchten Variablen werden in der Anfrage mit ‘?’ gesetzt, und als <sql:param ... > wie im Beispiel angegeben:

```
<sql:query var="bereitsAngemeldet">

    SELECT * FROM anmeldeliste WHERE anmeld_nachname = ? AND
anmeld_vorname = ? AND anmeld_geb = TO_DATE(?,'DD.MM.YYYY')

  <sql:param value="${param.nachname}" />
```

```
  <sql:param value="${param.vorname}" />
  <sql:param value="${geburtsdatum}" />
</sql:query>
```

Mit einem <sql:update> Statement, können nun die, durch den Benutzer angegebenen Werte hinzugefügt werden:

```
<sql:update>
     INSERT INTO anmeldeliste
    (anmeld_id, anmeld_nachname, anmeld_vorname, anmeld_geb, anmeld_matnr, anmeld_stud_kuerzel,
    anmeld_strasse, anmeld_land, anmeld_plz, anmeld_ort, anmeld_tel, anmeld_email, anmeld_datum)
    VALUES (anmeldeliste_seq.nextval, ?,
?,TO_DATE(?,'DD.MM.YYYY'), ?, ?, ?, ?, ?, ?, ?, ?, ?, SYSDATE)
     <sql:param value="${param.nachname}" />
     <sql:param value="${param.vorname}" />
     <sql:param value="${geburtsdatum}" />
     <sql:param value="${param.matnr}" />
     <sql:param value="${param.studiengang}" />
     <sql:param value="${param.strasse}" />
     <sql:param value="${param.land}" />
     <sql:param value="${param.plz}" />
     <sql:param value="${param.stadt}" />
     <sql:param value="${param.tel}" />
     <sql:param value="${param.mail}" />
</sql:update>
```

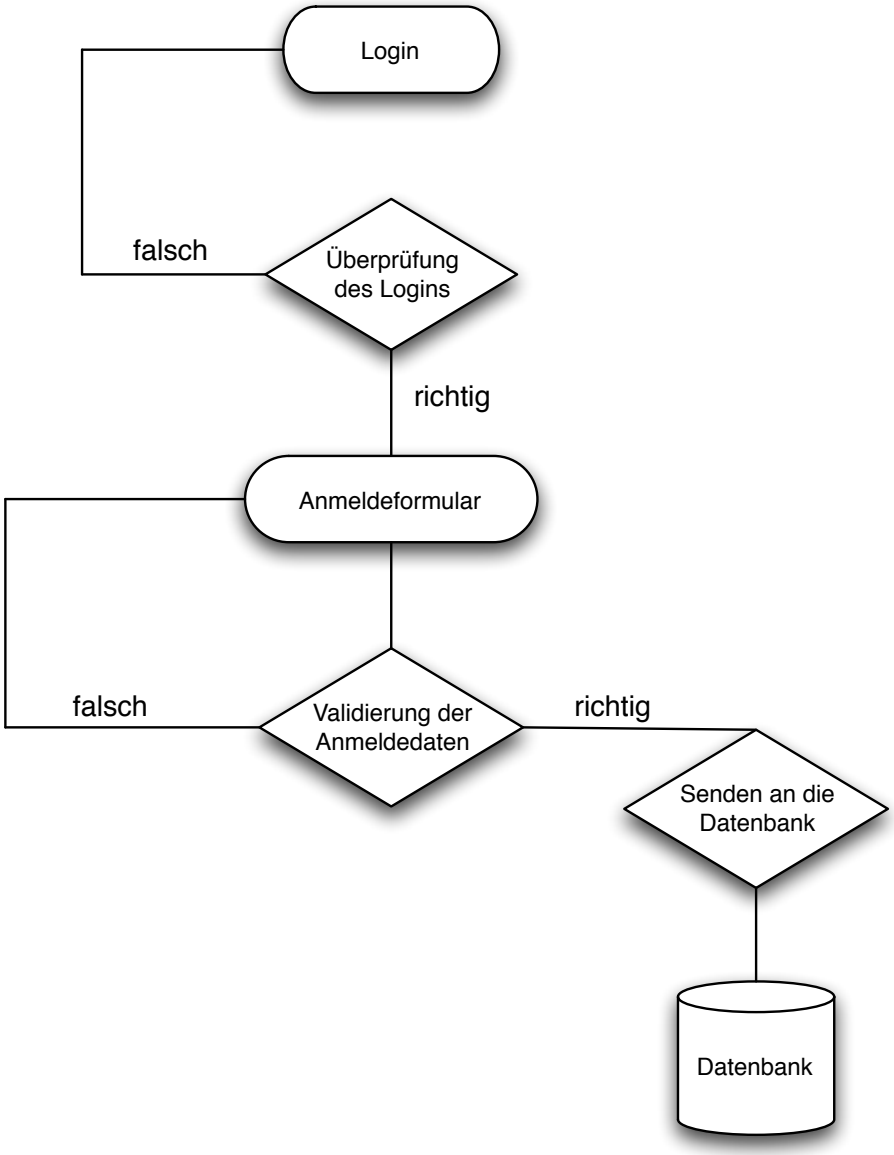
Mit der

```
<c:redirect url="anmeldung_XXXXXX.jsp" >
```

kann nun dem User angekündigt werden, ob die Anmeldung erfolgreich anmeldung_erfolg.jsp oder fehlerhaft (bei schon einem existierendem User) war. Hierzu werden drei Parameter in Bezug genommen:

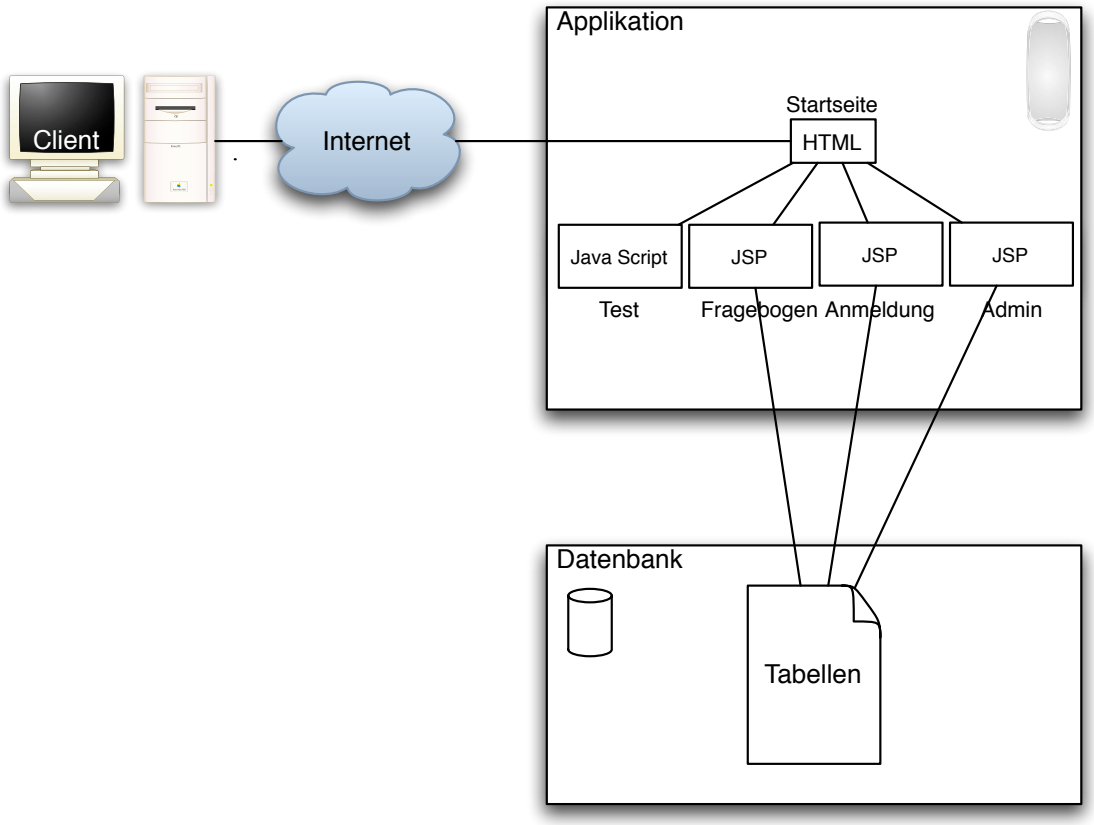
- Vorname
- Name
- Geburtsdatum

Nach dem selben Schema funktionieren alle anderen Funktionen die in dieser Anwendung mit eingebettet wurden.



Infrastruktur der Anwendung

Folgende Graphik soll die Infrastruktur des Online Anmeldetools abbilden.



Flexibilität der Anwendung

Die Datenbank wurde so konzipiert, dass Einträge von verschiedenen Jahrgängen trennbar ist. Es muss also nicht jedes Jahr die Tabellen gelöscht werden, stattdessen kann die Anmeldung sorglos fortgesetzt werden. Jede Tabelle (die mit den laufenden Einträgen der Vorkursteilnehmer in Beziehung steht) besitzt eine Spalte vom Typ 'date' die das Datum des Eintrages festhält. Daher reicht für die nächsten Jahrgänge jeweils immer eine Datenabfrage vom Typ:

```
SELECT alle_user FROM tabelle_deranmeldungen WHERE anmelde_datum
> 2005 AND (...)
```

Da auch eine Tabelle 'fragebogen_fragen' vorgesehen wurde, besteht die Möglichkeit (nach entsprechender Anpassung der JSP Dateien) die Fragen des Fragebogens zu ändern, zusätzliche Fragen hinzufügen oder umgekehrt.

Ausser der Datenbank, ist auch eine Flexibilität des Designs gegeben, da dieses ausschliesslich mit CSS beschrieben wurde. Näheres wurde aber schon unter dem Kapitel ‘HTML/CSS’ beschrieben.

Die Praxis

Der Enwicklungszeitrau: Vorbereitung und Evaluation

Die praktische Anwendung des Matheportals fing schon sehr früh mit der Implementierung an. Über den ganzen Entwicklungszeitraum (Februar/März 2005 - Juni/Juli 2005) waren potenzielle Benutzer mit engagiert und haben unser System am laufenden getestet und uns zur Korrekturen veranlasst. Dies hatte den Vorteil, dass wir im eigentlichen Benutzungszeitraum (dem Anmeldezeitraum) durch die Vorkursteilnehmer nur sehr geringe Fehler aufgetreten sind und somit grosse Schäden der Datenbank und sonstiger Struktur vermieden werden konnten.

Der Anmeldungszeitraum

In der Zeit der Anmeldungen haben uns keine negativen oder sonstige E-Mails von den Vorkursteilnehmern erreicht. Nach einer mündlichen Umfrage in den Klassenräumen wurde sie eher Begrüsst. Die Teilnehmer machten die Organisatoren des Vorkurses lediglich darauf aufmerksam, dass eine E-Mail-Bestätigung nach einer erfolgreichen Anmeldung wünschenswert wäre. Dies werden wir in die TODO Liste aufnehmen (War auch im Konzept nicht eingeplant- als weitergehendes Projekt realisierbar).
Von der Seite der Vorkurs-Organisatoren (und Dozenten) kam auch lediglich die Anmerkung, dass die Sortierung in den Tabellen falsch ist- sie entspricht nicht dem Schema: TT/MM/JJJJ.

Die Administrierung

Wo befindet sich das Matheportal?

Das Matheportal befindet sich zurzeit (stand März 2006) unter der Adresse:

- http://oras1.gm.fh-koeln.de:8000/mathevorkurs/index.html

Wie wird das Portal Administriert?

Um das Matheportal zu administrieren sind (für die Einfachen Funktionen) keine Programmierkentnisse nötig. Unter der Angegebenen Internetadresse befindet sich ein sogenannter „Admin-Bereich“. Dadrunter können, vorerst simple, aber die nötigsten Einstellungen vorgenommen werden um das Portal zu verwalten. Dazu gehören unter anderem:

- Freischaltung/Sperrung der Anmeldeseite
- Auswertung der Fragebogenumfrage
- Ändern von Passwortern der Benutzer
- Löschen/Hinzufügen von Teilnehmern
- Ausgabe von angemeldeten Benutzern
- Export der Ausgabe in Exel u.a. Formate

Diese Version des Matheportals ist noch „mager“ ausgestattet. Es fehlen einige Basisfunktionen, die wir aber für dieses QQ Projekt nicht eingeplant haben, somit aber anderen Kommilitonen die Möglichkeit geben dieses Portal zu erweitern und somit einen QQ2 Nachweis zu erwerben.

Erweiterung des Portals oder Ändern/Austauschen von Dateien

Auf den SSH-Server kommt man mit einem SSH Client (z.B. WinSCP für Windows, Terminal für Linux/OS X, oder Transmitt für Mac OS X). Man verbindet sich mit folgendem Server:

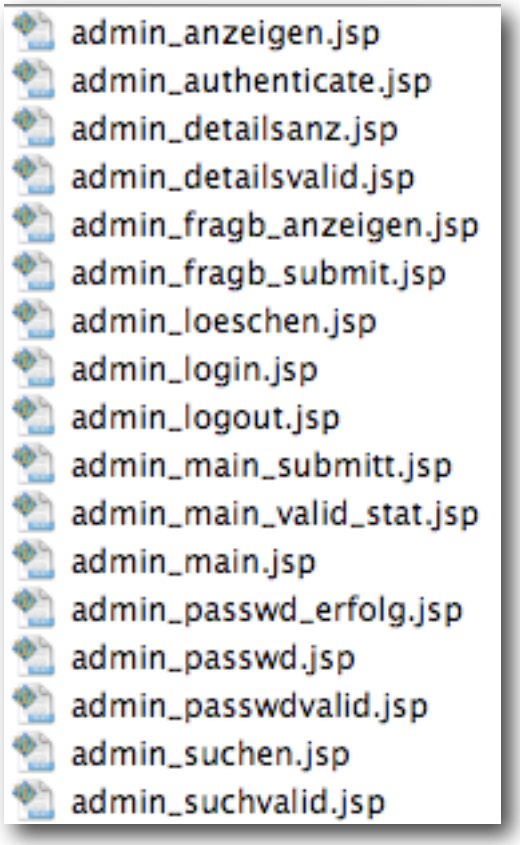
- oras1.gm.fh-koeln.de auf Port 22
- Benutzernamen und Password bitte beim Herrn Linke erfragen (linke@gm.fh-koeln.de)

nach einer erfolgreichen Verbindung liegen im Ordner

- tomcat/webapps/mathevorkurs/

Es soll hier ein kleiner Einblick über die Datenstruktur und eine Anleitung für verschiedene Anwendungsfälle beschrieben werden.

Dazu betrachten wir uns in erster Linie die .JSP Dateien die mit „admin_“ anfangen. Dazu gehören bis jetzt:



Anwendungsfall: Alle Anmeldungen Anzeigen

Wird z.B. der erste Menüpunkt aktiviert: **Alle Anmeldungen anzeigen** so wird die Datei **admin_anzeigen** angesprochen in der die Entsprechenden SELECT-Statements ausgeführt werden und die Tabelle mit den Anmeldungen angezeigt wird. Soll in der Darstellung oder in der SELECT-Abfrage hier was geändert werden, so muss die Datei:

- admin_anzeigen.jsp

geändert werden. In dieser Datei könnte auch ein Zusatz-SELECT-Statement für die Abfrage der Anmeldung nach ihren Anmeldejahren erfolgen.

Anwendungsfall: Details für eine bestimmte Anmeldung zeigen

In diesem Fall müssen die Dateien

- admin_detailsanz.jsp

- admin_detailsvalid.jsp

beachtet werden. In diesem Beispiel wurde darauf geachtet die SELECT-Abfrage von der Darstellung zu trennen. Das SELECT-Statement wird in der

- admin_detailsvalid.jsp

aufgerufen und die Darstellung mittels

- admin_detailsanz.jsp

ausgegeben.

Anwendungsfall: Fragebogenauswertung anzeigen

Analog zu der Detailanzeige werden hier auch eine „Anfrage-“ und „Darstellungsdatei“ betrachtet:

- admin_frag_anzeigen.jsp
- admin_fragb_submit.jsp

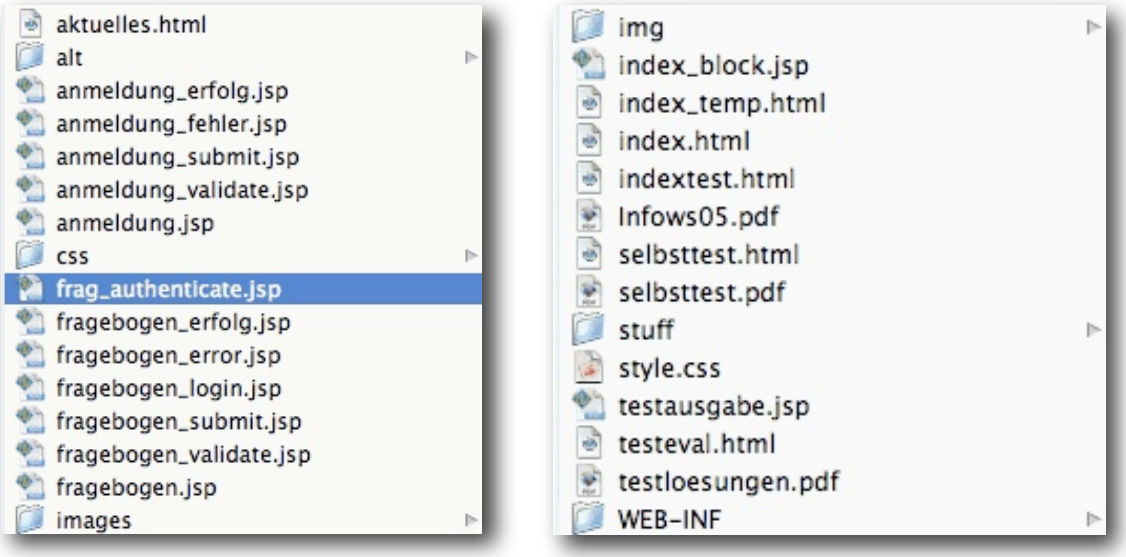
Durch Manipulation der

- admin_fragb_submit.jsp

besteht die Möglichkeit die einzelnen Anfragen so zu ändern dass sie z.B. nur Auswertungen aus dem Jahr 2005 anzeigen.

Anzeige und Manipulationsmöglichkeiten auf der Benutzerebene

Um das Portal auch auf der Benutzerseite ändern zu können (Z.b. Anmeldungsformular ändern, oder neue .pdf Dateien hochladen etc. müssen die restlichen Dateien im Paket beachtet werden.



Folgende Ordner sind für diese Änderungen relevant:

- *css/* und *img/* beinhalten Dateien die Notwendig sind für die Tabellarische Ausgabe des „display"-Taglibs (JSP)
- *images/* enthält alle Bilder die im ganzen Portal verwendet werden, Logo u.a.
- *stuff/* sind zusätzliche Module die heruntergeladen wurden (JSP)
- *style.css* ist die CSS-Beschreibung für das HTML Layout
- *aktuelles.html, index.html* sind HTML Ausgabedateien die nicht dynamisch verarbeitet oder aufbereitet werden
- *selbsttest.html, testeval.html* sind die dynamisch verarbeiteten JavaScript Dateien für den Eignungstest
- **.pdf* - alle Zusatzmaterialien im .pdf-Format die auf der Homepage angeboten werden.
- anmeldung_XXX.jsp sind Dateien die für die dynamische Verarbeitung der Anmeldeinformationen verantwortlich sind (Absenden an die Datenbank, speichern, etc.)
- fragebogen_XXX.jsp ermöglichen die Manipulation des Fragebogens (Ändern der Fragen, schreiben in die Datenbank, etc.)

Anwendungsfall: Das Uploaden und Veröffentlichen einer .pdf Datei (oder ersetzen einer Alten)

In so einem Fall ist es nur erforderlich die entsprechende Datei hochzuladen und auf der entsprechenden html Seite zu verlinken (z.B. aktuelles.html)

Anwendungsfall: Hochladen neuer .jsp Dateien

In so einem Fall ist es auch nur erforderlich die neue .jsp Datei hochuzladen und entsprechend in die ganze Struktur einzubinden. Möglicherweise reicht es einen Link in die Darstel-

lungsdateien (z.B. anzeigen.jsp, u.a.) hinzuzufügen. Bei komplexeren Aufgaben werden wahrscheinlich die Einbettung neuer if- oder while-Schleifen o.a. nötig sein.

Anwendungsfall: Fragen im Fragebogen hizufügen oder löschen/ändern

Dieser Vorgang bezieht sich nur auf die Datei

- fragebogen.jsp

Hier werden die Fragen eingegeben.

Achtung: zurzeit werden die Fragen noch manuell in die Datei eingefügt. Es ist eine Fragebogen-Tabelle in der Datenbank angelegt, wird aber noch nicht benutzt. Es empfiehlt sich von dieser Tabelle in der nächsten Iteration der Implementierung Gebrauch zu machen, da das Einfügen oder Löschen von Fragen dann wesentlich einfacher ist.

Abgrenzung zum WPF-Portal

Was grenzt das Matheanmeldeportal vom Wahlpflichtfachanmeldeportal ab?

Beide Portale verwenden die selbe Technik: JSP, Datenbanken, HTML/CSS. Dennoch ist in folgenden Punkten die Abgrenzung vorzunehmen:

- Entlastung der Studentischen Hilfskräfte
- Zielgruppe
- Struktur der Datenbank
- Darstellung
- Einstufungstest
- Fragebogen und Fragebogenauswertung
- Suchen von Teilnehmern

Das primäre Ziel war die Entlastung der Stundentischen Hilfskräfte. Das verschicken und beantworten aller E-Mails zum Anmeldezeitraum an die Vorkursteilnehmer hat jedes Jahr einen grossen Aufwand für die beteiligten Organisatoren des Vorkurses bedeutet. Es wurden darüber hinaus Exel-Tabellen erstellt in denen alle angemeldeten Vorkursteilnehmer manuell eingetragen wurden. Diese umfassten eine Zahl von ca. 200 Teilnehmer. Unser Mathematikportal ermöglicht in der Zukunft eine automatische Anmeldung, Tabellenerstellung für die Dozenten und studentischen Hilfskräfte, sodass ein grosser Teil des organisatorischen Aufwandes wegfällt.

Darüber hinaus besitzt das Mathematikvorkursportal einen Fragebogen der durch die Teilnehmer des Vorkurses, nach dessen Abschluss ausfüllen können. Es zeigte sich jedoch, dass dieses von den Teilnehmern nicht genutzt wurde. Es gab leider lediglich nur 15 Einträge (von 200 Teilnehmern- 2005). So eine geringe Anzahl von ausgefüllten Fragebögen kann keine gute Aussage bilden, somit ist die Auswertung aus der statistischen Sicht her nicht benutzbar. Für die Zukunft ist es deshalb ratsam diese Onlinemöglichkeit als eine Alternative zu den bestehenden Papierfragebögen anzubieten.

Ein weiterer Faktor der das Mathematikportal abgrenzt ist der Online-Einstufungstest, der sich in seiner Praxis ganz gut bewährt hat. Die Studenten hatten die Möglichkeit Ihre Kennt-

nisse über Ihren Mathematikwissenstand direkt online zu erfahren und darauf hin direkt eine Empfehlung bekommen ob der/die jeweilige Teilnehmer(in) sich anmelden soll oder nicht.

Kleinere Faktoren die oben Aufgelistet worden sind u.a. die Darstellung (Präsentationsschicht des Portals), die Datenbankmodelierung und die JSP-Umsetzung.

Quellenangabe

Verwendete Literatur

Titel: JavaServer Pages
Verlag: O'Reilly, August 2002
ISBN: 0-596-00317-X

Aus dem Internet

http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/
http://java.sun.com/webservices/downloads/install-unix.html
http://www.jsp-develop.de/links/
http://java.sun.com/developer/technicalArticles/Programming/jsp/
http://www.databasejournal.com/features/oracle/article.php/3344871
http://www.akadia.com/services/ora_date_handling.html

Verwendung von früheren Services der Fachhochschule Köln (unter Verwendung von JSP)

Als Beispiel zur technischen Realisierung des Projektes diente uns das WPF-Anmeldeportal entwickelt von **Thuan Truong** (11030237, thuan@truong-clan.de). Sein Portal war eine sehr grosse und hilfreiche Unterstützung der Mathematikportals. Einige „Codezeilen" (wie z.B. die tabellarische Darstellung der Angemeldeten Benutzer) haben wir mit kleinen Anpassungen übernommen. An dieser Stelle wollen wir uns auch sehr herzlich bei **Thuan** bedanken!

Arbeits- und Zeiteinteilung

Natürlich haben sich einige Dinge bei der Arbeit überschnitten und eine klare Teilung der Aufgabenbereiche war nur teilweise möglich (und von uns auch gar nicht erwünscht). Allerdings lässt sich der Aufgabenbereich grob in folgende Bereiche aufteilen:

HTML/CSS

Norma Yvonne Sappelt hat sich grundliegend um die Gestaltung von der HTML Seite gekümmert. Unter der Verwendung der CSS wurde dabei auf eine klare Abtrennung des Designs vom Inhalt vorgenommen.

Für das erstellen des HTML und CSS Codes hat Norma Yvonne Sappelt ein Stundenaufwand von ca. 6 H erbracht.

Programmierer	Stundenanzahl
Norma-Yvonne Sappelt	6
Jan Kus	0

Datenbank

Jan Kus und Norma Yvonne Sappelt beschäftigten sich mit der Entwicklung von Datenbanken.

Die Datenbank war der Aufwendigste Teil dieser Aufgabe. Es beanspruchte vor allem gute Vorbereitung vor der technischen Umsetzung, sodass eine, für längere Jahre, brauchbare Anwendung entsteht. Insgesamt wurden ca. 20 H für die Vorbereitung und 35 H für die Umsetzung der Datenbank, pro Programmierer gebraucht.

Programmierer	Stundenanzahl
Norma-Yvonne Sappelt	40
Jan Kus	40

JSP

Die JSP Seiten von Norma Yvonne Sapplet und Jan Kus erstellt. Diese Aufgabe war nach der Datenbank die Zweitauweitaufwändigste, aufgrund des Versuches die Funktion vom View zu trennen, Hierfür wurden von Jan Kus ca. 30 H gebraucht, Norma-Yvonne Sappelt hat ca. 15 H Zeitaufwand in diesen Teil der Aufgabe aufgewendet.

Um die ständige Pflege der Updates beim Administrator des Servers der Fachhochschule Köln hat sich Jan Kus gekümmert, woraus noch ein zusätzlicher Aufwand von ca. 8 Stunden aufkam.

Programmierer	Stundenanzahl
Norma-Yvonne Sappelt	25
Jan Kus	45

JavaScript

Norma Yvonne Sappelt entwickelte den JavaScript basierten Selbsteinstufungstest.

Programmierer	Stundenanzahl
Norma-Yvonne Sappelt	10
Jan Kus	0

Gesamtanzahl der Stunden

In der Summe haben die einzelnen Personen folgende Anzahl an Arbeitsaufwand gebraucht:

Programmierer	Stundenanzahl
Norma-Yvonne Sappelt	81
Jan Kus	85

9