INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen

Oettingenstraße 67 D–80538 München

Ludwig
Maximilians-
Universität
München

Einsatz elektronischer Medien für den Übungsbetrieb

Georg Schneemayer

Projektarbeit im Rahmen des Fortgeschrittenenpraktikums

Abgabe der Arbeit:	26. Februar 2001
Aufgabensteller:	Prof. Dr. François Bry
Betreuer:	Dr. Norbert Eisinger

Einsatz elektronischer Medien für den Übungsbetrieb: Projektarbeit

von Georg Schneemayer

Inhaltsverzeichnis

Abbildungsverzeichnis

Beispiele

Motivation der Projektarbeit

Die stark gestiegenen Studentenzahlen in der "Informatik I" Vorlesung im Wintersemester 2000/2001 führen zu einem erheblichen organisatorischen Aufwand. Insbesondere die Abgabe und die Bewertung der Übungsaufgaben im Rahmen des Übungsbetriebs ist bei hohen Studentenzahlen kaum in bisheriger Weise handhabbar. Die Lösung dieses Problems liegt in der zumindest teilweisen Automatisierung mit Hilfe eines Systems von Programmen, die Abgabe, Test und Bewertung sowie Verwaltung der Studenten elektronisch ermöglichen und somit den organisatorischen Aufwand reduzieren.

Im folgenden wird dieses System und seine Bedienung durch die verschiedenen Benutzer-Gruppen beschrieben. Es wird eine Installationsanweisung sowie Hinweise auf notwendige Anpassungen bei der Verwendung für andere Vorlesungen gegeben.

Kapitel 1. Einführung und Überblick

In diesem ersten Abschnitt wird die Problemstellung formuliert, die der Projektarbeit zugrunde liegt. Es wird auf die Ziele der Arbeit eingegangen und es werden die Entscheidungen diskutiert, die während des Designs und im Laufe der Implementierung getroffen worden sind. Zudem wird ein kurzer Überblick über die Realisierung des Systems gegeben.

1.1. Ausgangslage

Bei großen Studentenzahlen steigt der Zeitbedarf für die im Übungsbetrieb anfallenden verwaltungstechnischen und organisatorischen Aufgaben und führt damit zum einen zu einer großen Belastung für die Organisation der Veranstaltung, zum anderen bleibt damit aber auch weniger Zeit für die Betreuung der Studenten.

Um den Aufwand für die Durchführung des Übungsbetriebs zu verringern bietet sich der Einsatz elektronischer Hilfsmittel an. Im folgenden werden die Möglichkeiten aufgezeigt, die die Durchführung mit elektronische Unterstützung im Vergleich zur herkömlichen Durchführung bietet.

1.1.1. Verwaltung der Daten

Eine elektronische Verwaltung der bei der Durchführung einer Veranstaltung anfallenden Daten kann den Aufwand verringern helfen.

Daten die im Übungsbetrieb anfallen sind u.a.:

- Daten der Studenten wie Name, Studienfach und Matrikelnummer
- Daten der Betreuer der Übungsgruppen (Übungsgruppen-Leiter und Korrektor)
- Informationen zu den einzelnen Übungsgruppen (u.a. Zeit, Ort und Betreuer)
- Punkte der Studenten bei den Übungsblättern und bei der Klausur

Liegen diese Daten in elektronischer Form vor, so lassen sich mit vergleichsweise wenig Aufwand die Daten auswerten. Interessant ist dabei z.B. der Zusammenhang zwischen Klausur-Punktzahl und Punktzahl auf den Übungsblättern.

Die elektronische Vorhaltung der Daten stellt jedoch per se noch keine Arbeitserleichterung dar. Es muss auch eine effiziente Methode gefunden werden, wie diese Daten eingegeben, aktualisiert und verwaltet werden können.

1.1.2. Organisation der Übungsblatt-Abgabe

Bisher haben die Studenten die Übungsblätter auf Papier gelöst und sie durch Einwurf in einen Briefkasten für die jeweilige Vorlesung abgegeben. Dieser Briefkasten wird zu einem bestimmten

Zeitpunkt geleert, und die darin enthaltenen Lösungen an die Korrektoren verteilt.

Wünschenswert bei dieser Verteilung ist, dass die einzelnen Korrektoren bei jedem Übungsblatt die selben Studenten korrigieren. Dies setzt jedoch eine zeitaufwändige Zuordnung der Lösungen an die Korrektoren voraus.

Bei einer elektronischen Abgabe der Lösungen durch die Studenten kann die Verteilung der Übungsblätter an die Korrektoren automatisch durchgeführt werden. Da die meisten Aufgaben gerade in "Informatik I" am Computer zu lösen sind, bedeutet die Abgabe in am Computer auch für die Studenten weniger Aufwand.

Ein Nachteil der elektronischen Abgabe ist jedoch, dass sie nur für bestimmte Aufgabentypen in Frage kommt. So ist sie z.B. für mathematische Texte ungeeignet und wäre damit unter anderem für die Vorlesung "Informatik IV" nicht geeignet.

1.1.3. Korrektur der Lösungen

Bei Abgabe auf Papier ist die Korrektur von Programmieraufgaben schwer durchzuführen. Schon die Durchführung einer einfachen Syntaxüberprüfung ist von Hand sehr aufwändig. Das abgegebene Programm kann nicht automatisch oder zumindest mit Computerunterstützung getestet werden.

Liegen dagegen die Aufgaben nicht auf Papier sondern in elektronischer Form vor, können sie leicht auf korrekte Syntax überprüft und, mit geeigneten Hilfsmitteln, automatisch getestet werden. Auch die Wiederverwendung von Korrektur-Kommentaren bei gleichartigen Fehlern wird erleichtert.

1.1.4. Rückgabe der korrigierten Lösungen an die Studenten

Ebenso wie die Abgabe der Übungsblätter durch die Studenten und die Verteilung der Übungsblätter an die Korrektoren stellt auch die Rückgabe der Übungsblätter ein organisatorisches Problem dar.

Beim bisherigen Vorgehen lässt der Korrektor die korrigierten Übungsblätter dem Übungsgruppen-Leiter zukommen, dieser sorgt dann für die Rückgabe an die Studenten in den Übungsgruppen. Nicht in der Übung anwesende Studenten muss entweder in der kommenden Übung oder auf andere Art und Weise ermöglicht werden, ihre korrigierten Übungs-Blätter zurück zu erhalten.

Liegen hingegen die korrigierten Aufgaben in elektronischer Form vor, so können sie den Studenten auch in elektronischer Form (z.B. in einer E-Mail) zurückgegeben werden. Der Aufwand für die Rückgabe der korrigierten Lösungen an die Studenten entfällt dadurch vollständig. Auch das Ausdrucken auf Papier kann durch die elektronische Rückgabe vollständig entfallen.

1.1.5. Verwaltung der Punktzahlen

Eine bestimmte Punktzahl auf den Übungsblättern ist Voraussetzung für die Teilnahme an den Klausuren. Die Punktzahlen werden von den Korrektoren vergeben und bisher auch von diesen verwaltet. Hinzu kam ein regelmäßiger Übertrag der Punkte an eine zentrale Stelle.

Die Punktzahlen mussten von Hand in den Computer eingegeben werden, um automatisch Zulassungslisten u.ä. für die einzelnen Klausuren erzeugen zu können.

Liegen die Informationen über die Studenten und die Übungsblätter bereits in elektronischer Form vor, so bietet sich an, auch die erreichten Punktzahlen elektronisch zu verwalten. Die doppelte Vorhaltung der Punktzahlen bei den Korrektoren und an einer zentralen Stelle könnte somit entfallen.

1.2. Anforderungen und Ziele

Oberstes Ziel des Systems ist die Erleichterung der Arbeitsabläufe im Übungsbetrieb. Daher muss der Aufwand für den Betrieb des Systems so niedrig wie möglich gehalten werden. Um dies zu gewährleisten müssen die folgenden Punkte eingehalten werden.

1.2.1. Geringer Aufwand bei regulärem Betrieb

Der reguläre Betrieb des Systems soll möglichst wenig Aufwand verursachen. Das System soll einmal zu Beginn des Semesters eingerichtet werden und danach selbst möglichst wenig Aufwand verursachen. Gleichzeitig sollen leicht Änderungen, z.B. bei den Aufgaben bestimmter Übungsblätter, vorgenommen werden können.

1.2.2. Sonderfälle

Bei großen Studentenzahlen wie sie in "Informatik I" im Wintersemester 2000/2001 vorkommen, verursachen Sonderfälle, wie z.B. Krankheit eines Studenten, der dadurch keine Übungsblätter abgeben konnte, einen erheblichen zusätzlichen Zeitaufwand. Ist das System darauf nicht vorbereitet, kommt noch zusätzlicher Aufwand hinzu.

Vermeidung von Sonderfällen. Ziel ist es daher solche Sonderfälle möglichst zu vermeiden bzw. sie wie reguläre Fälle behandeln zu können.

Hilfsmittel zur Behandlung von Sonderfällen. Geeignete Hilfsmittel zur Behandlung von häufig auftretenden Sonderfällen, wie z.B. Nach-Korrektur von einzelnen Aufgaben, sollen es ermöglichen, diese Fälle mit wenig Zeitaufwand zu erledigen.

manuelle Eingriffsmöglichkeit. Da nicht alle Sonderfälle vorhergesehen werden können, ist es wichtig, auch manuell in das System eingreifen zu können. Aus System-Sicht wird dabei bspw. ein Sonderfall zu einem regulären Fall.

1.2.3. Minimierung von Bedienfehlern/-schwierigkeiten

Insbesondere Bedienfehler und -schwierigkeiten auf Seiten der Studenten verursachen einen erheblichen Betreuungsaufwand. Daher muss es Ziel sein, die Bedienung einfach und klar zu gestalten, damit möglichst wenig Missverständnisse auftreten.

Verwendung bekannter Bedienkonzepte. Die Verwendung von den Benutzern bekannten Bedienkonzepten (wie z.B. der Umgang mit HTML-Formularen und E-Mail) erleichtert zudem die Benutzung.

Abfangen von Bedienfehlern. Trotzdem auftretende Bedienfehler müssen abgefangen werden und der Benutzer auf seinen Fehler hingewiesen werden. Dazu gehört unter anderem auch eine Überprüfung der eingegebenen Daten.

1.2.4. Geringe Anforderungen an die Systeme der Studenten und Korrektoren

Das System soll ohne Probleme mit den Rechnern der Korrektoren und der Studenten einsetzbar sein. Es muss also auf allen gängigen Betriebs-Systemen laufen und dabei keine besonderen Software-Voraussetzungen oder zusätzliche Installation von Software voraus setzen. Es soll mit Standardsoftware benutzbar sein.

1.2.5. Sicherheit

Die Sicherheit des Systems umfasst nicht zuletzt aufgrund der großen Benutzer-Gruppe viele Aspekte.

Sicherheit vor Datenverlust. Einer davon ist die Sicherheit der gespeicherten Daten vor Verlust. Dies kann durch regelmäßige Backups geschehen.

Ablauf-Sicherheit. Ein weiterer wichtiger Punkt ist die Ablauf-Sicherheit des Systems. Das System darf andere Prozesse auf dem Server nicht beeinflussen und die Betriebs-System-Stabilität nicht beeinflussen. Dies kann u.a. durch Beschränkung der zur Verfügung stehenden Ressourcen und eingeschränkter Zugriffsrechte bei bestimmten Arbeitsabläufen geschehen. Dieser Punkt gewinnt an Wichtigkeit, da bei automatischen Tests die Programme der Studenten auf dem Server ausgeführt werden müssen.

Sicherheit vor unberechtigten Zugriff. Die Sicherheit der Daten und abgegebenen Lösungen der Benutzer ist ein weiterer Aspekt. Dazu gehört, dass Unbefugte nicht die Daten anderer Benutzer einsehen können und insbesondere nicht deren Passwörter, die zur Authentifizierung der Benutzer durch das System benötigt werden, ausspionieren können.

Sicherheit vor Manipulation. Auf Sicherheit vor Manipulation der Daten ist insbesondere bei den Punkte-Ständen zu achten. Diese dürfen nicht von unberechtigten Personen verändert werden können.

1.2.6. Flexibilität

Das System muss ausreichend flexibel sein, um auch künftig bei anderen Vorlesungen mit anderen Anforderungen verwendet werden zu können. Dazu ist ein modularer Aufbau notwendig, der eine leichte Erweiterbarkeit gewährleistet.

1.3. Entwurfs-Entscheidungen

Die für die Realisierung des Systems getroffenen Entwurfs-Entscheidungen werden in den folgenden Abschnitten erläutert. Dabei werden auch alternative Lösungen aufgezeigt und auf ihre Vor- und Nachteile der eingegangen.

1.3.1. Abgabe - HTML-Formular oder Home-Verzeichnis

Anforderungen an die Abgabe der Lösung sind:

- Für die Studenten leicht zu handhaben
- Eindeutige Identifikation des Studenten bei Abgabe
- Möglichkeit der Abgabe der Lösungen nur im vorgesehen Zeitraum
- Abgabe-Zeitraum muss bei Bedarf (z.B. aufgrund technischer Probleme) leicht verlängert werden können

1.3.1.1. Abgabe über das Home-Verzeichnis

Bei dieser Abgabe-Variante hat jeder Student in seinem Home-Verzeichnis ein Abgabe-Verzeichnis mit Unterverzeichnissen für die einzelnen Übungsblätter. Die Rechte dieses Verzeichnisses sind so gesetzt, dass andere Studenten dieses Verzeichnis nicht einsehen können.

Die Abgabe erfolgt, indem der Student die Lösungen in das Verzeichnis für das jeweilige Übungsblatt kopiert. Am Ende des Bearbeitungs-Zeitraumes des Übungsblattes wird ein Programm gestartet, welches die Lösungen aus den Verzeichnissen der Studenten abholt.

1.3.1.1.1. Vorteile

eindeutige Zuordnung. Die abgegebene Lösung kann eindeutig dem Besitzer des Home-Verzeichnisses zugeordnet werden.

Kontrolle über das Abholen der Lösung. Das Abholen der Lösung kann bei technischen Problemen ohne weiteres wiederholt werden. Das Abholen der Lösung ist nicht an eine bestimmte Zeit gebunden und nicht an die Funktionsfähigkeit bestimmter Systeme zu bestimmten Zeiten.

Testen auf CIP-Rechnern. Da die Studenten die Lösungen auf einem CIP-Rechner ablegen, ist die Chance größer, dass die Studenten die Aufgaben auch an den CIP-Pool Rechnern getestet haben. Die Gefahr, dass die abgegebenen Programme bei den Studenten zu Hause funktionieren, nicht jedoch auf den CIP-Rechnern, ist dadurch geringer.

1.3.1.1.2. Nachteile

komplizierter für die Studenten. Die Studenten müssen eventuell die Dateien extra auf die CIP-Rechner kopieren, falls sie die Aufgaben zu Hause gelöst haben.

erleichtert Abschreiben. Wenn ein Student die Aufgaben am CIP-Pool löst und abspeichert, können andere Studenten die Aufgaben einsehen, falls der Student die Rechte in seinem Home-Verzeichnis nicht beschränkt hat. Eine Beschränkung der Zugriffsrechte des Abgabe-Verzeichnisses löst das Problem nicht, da die Studenten in diesem Verzeichnis nur die fertigen Lösung ablegen, ihre Arbeitskopien jedoch an beliebiger Stelle in ihrem Verzeichnis ablegen.

keine Bestätigung der Abgabe. Die Studenten erhalten keine unmittelbare Bestätigung der Abgabe, die Bestätigung kann erst erfolgen, wenn das Abhol-Skript die Lösungen aus dem Abgabe-Verzeichnis kopiert.

kein Test der Abgabe. Der Test, ob die Abgabe angenommen wird, erfolgt erst bei der Abholung. Den Studenten kann zwar ermöglicht werden diesen Test selbst auch durchzuführen, der Test des Studenten und der eigentliche Test beim Abholen der Lösungen bleibt jedoch verschieden.

1.3.1.2. Abgabe über HTML-Formular

Die Studenten geben die Lösungen über ein HTML Formular ab.

1.3.1.2.1. Vorteile

einfacher für die Studenten. Die Studenten können die Abgabe von jedem Rechner mit Internet-Zugang mit Hilfe eines Browsers durchführen. Auch ist die Bedienung eines HTML-Formulars den meisten Studenten vertrauter als Datei-Operationen auf einem Linux-Rechner.

Bestätigung der Abgabe. Die Studenten erhalten eine unmittelbare Bestätigung der Abgabe. Die Abgabe wird geprüft, und der Student bekommt sofort eine Rückmeldung, ob die Abgabe erfolgreich war.

keine Konflikte bei der Team-Zuordnung. Es wird jedem Team nur eine Abgabe ermöglicht. Versucht eines der Team-Mitglieder eine weitere Abgabe, bekommt es eine Meldung, dass die Abgabe bereits erfolgt ist.

1.3.1.2.2. Nachteile

eindeutige Zuordnung. Die eindeutige Zuordnung kann nur durch ein Passwort geschehen. Die Ausgabe und Verwaltung eigener Passwörter ist jedoch sehr aufwändig. Daher bleibt die Möglichkeit, die CIP-Pool Passwörter zu benutzen. Die Passwörter werden dann jedoch über das Internet übertragen. Für diese Übertragung muss eine SSL-Verbindung zur Verfügung gestellt werden. Die Überprüfung des Passworts erfordert ein Programm mit Lese-Rechten für die Passwort-Datei /etc/shadow . Lese-Rechte für diese Datei hat nur root und Mitglieder der shadow Gruppe. Das CGI-Programm, welches das Formular verarbeitet, muss also ein Programm mit sgid shadow aufrufen.

technische Probleme. Falls die Studenten aufgrund von technischen Problemen nicht abgeben können, muss, falls dies kurz vor dem Abgabe-Termin stattfindet, der Abgabe-Termin verschoben werden, um den Studenten ausreichend Zeit zur Abgabe zu geben. Aus diesem Grund muss die Verlängerung des Abgabe-Zeitraums einfach zu handhaben sein.

Verfügbarkeit des Servers. Die Verfügbarkeit des Servers muss gewährleistet sein. Dies gilt insbesondere zu der Spitzenzeit der Belastung kurz vor Ende des Abgabezeitraums.

wöchentliche Erstellung einer Abgabe-Seite. Die Abgabe-Seiten sind für die einzelnen Übungsblätter unterschiedlich. Sie müssen daher für jedes Übungsblatt neu erstellt werden. Die Erstellung der HTML-Seiten kann jedoch automatisiert werden.

1.3.2. Korrektur - nur elektronisch, oder auch auf Papier

Zu Beginn der Projektarbeit stand die Frage im Raum, ob eine nur elektronische Korrektur praktikabel sei, oder ob die Korrektoren nicht doch lieber auf Papier korrigieren wollen.

Eine Umfrage unter den Korrektoren zu Beginn der Projektarbeit ergab keine ausschließliche Präferenz für eine Korrektur auf Papier. Die Korrektur auf Papier würde für die Organisation einen wesentlich höheren Aufwand verursachen. Die Aufgaben würden anstatt den Korrektoren nur elektronisch zur Verfügung gestellt zu werden zusätzlich auch ausgedruckt werden und an die Korrektoren verteilt werden.

Diese könnten dann die Aufgaben auch auf Papier korrigieren und die korrigierten Aufgaben dann auf Papier den Studenten zurückgeben. Dies würde eine Verbindung des alten Systems ohne elektronischer Unterstützung und des neuen Systems bedeuten, wobei durch den zumindest teilweisen parallelen Betrieb beider Systeme der Aufwand tendenziell größer ist als bei nur einem System.

Aufgrund dieser Erwägungen wurde die Mischform, Korrektur auf Papier mit elektronischer Unterstützung, verworfen.

1.3.3. Verwaltung der Daten

Die im Laufe des Semesters anfallenden Daten wie z.B. die Punktestände der Studenten oder die Information, welche Studenten zusammen abgegeben haben, müssen in geeigneter Form gespeichert werden. Grundsätzlich kommen dazu zwei verschiedene Möglichkeiten in Frage, zum einen die Verwendung eines DBMS, zum anderen die Speicherung in Dateien.

1.3.3.1. Anforderungen an die Datenverwaltung

Die Entscheidung für eine der beiden Alternativen, DBMS oder Dateisystem, muss aufgrund der Anforderungen an die Datenverwaltung gefällt werden:

- **großer Datenumfang.** Die im Laufe des Semesters anfallenden Datenmengen müssen effizient verwaltet, durchsucht und ausgewertet werden können. werden können. Die abgespeicherten Informationen müssen miteinander verknüpft werden können, so dass auch Informationen wie die Punktestände aller Studenten einer bestimmten Übungsgruppe leicht extrahiert werden können.
- **nebenläufige Zugriffe.** Da mehrere Studenten über das Web-Formular gleichzeitig abgeben können, müssen auch nebenläufige Zugriffe (sowohl lesend als auch schreibend) auf die Daten möglich sein, ohne dass es dabei zu Inkonsistenzen kommt.
- **Sicherheit.** Die Sicherheit der Daten umfasst zum einen die Sicherheit vor Datenverlust als auch die Sicherheit vor unberechtigten Zugriff. Beide Aspekte müssen erfüllt sein.

1.3.3.2. Alternativen

Die Entscheidung zwischen den beiden alternativen, DBMS oder Dateisystem, fällt aufgrund der oben genannten Anforderungen klar zugunsten eines DBMS aus. Ein DBMS erfüllt alle oben genannten Anforderungen. Mit einer dateibasierten Datenhaltung müssten Anforderungen wie nebenläufige Zugriffe selbst implementiert werden, was zu einem erheblichen Mehraufwand bedeutet.

Die Entscheidung für das DBMS MySQL fiel aufgrund der Verfügbarkeit auf dem Server. Auch wenn MySQL nicht den vollen Sprachumfang von SQL unterstützt (MySQL kennt in der eingesetzten Version keine Transaktionsunterstützung und unterstützt auch keine Views), so werden doch alle benötigten Anforderungen erfüllt.

1.3.4. Programmiersprache zur Implementierung

Auch wenn die zur Implementierung verwendete Programmiersprache nicht zu den eigentlichen Entwurfs-Entscheidungen gehört, so hat doch die Wahl einer Programmiersprache direkt mit bereits getroffenen Entwurfs-Entscheidungen zu tun. Ist die Wahl für eine Programmiersprache schließlich getroffen, so beinflusst diese wiederum Implementierungs-Entscheidungen. Aus diesen Grunden wird die Wahl der Programmiersprache in diesem Abschnitt diskutiert.

1.3.4.1. Anforderungen an die Programmiersprache

Die Aufgaben des Systems bedingen eine Realisierung in einer Programmiersprache, die den folgenden Ansprüchen gerecht wird:

- **einfache Datenbank-Anbindung.** Die im Laufe des Semesters anfallenden Daten müssen in einer Datenbank verwaltet werden. Die Programmiersprache muss daher eine einfache Datenbank-Anbindung ermöglichen.
- **einfache Verarbeitung von XML-Dateien.** Die Informationen zu den Übungsblättern sollen in XML-Dateien abgelegt werden. Ein einfacher Zugriff in der Programmiersprache auf dieses Datenformat ist somit notwendig.
- **einfache Anbindung von externen Programmen.** Viele der Aufgaben des System bedingen externe Programme. Beispiele hierfür sind die Verwendung des Compilers für die bei den Übungsaufgaben verwendete Programmiersprache, die Ausführung der abgegebenen Programme zu Testzwecken, aber auch das Kopieren und Archivieren der Dateien. Die Implemetierungs-Programmiersprache muss also einen einfachen Mechanismus besitzen, mit externen Programmen über die Standard Ein- und Ausgabe-Kanäle zu kommunizieren.
- **reguläre Ausdrücke oder vergleichbares zur Analyse von Programm-Ausgaben.** Die Ausgabe externen Programmen ist oft nicht für die elektronische Weiterverarbeitung bestimmt, sondern an einen menschlichen Benutzer gerichtet. Um diese Ausgaben trotzdem elektronisch weiterverarbeiten zu können, sind reguläre Ausdrücke zur Interpretation der Ausgabe notwendig.
- **Eignung für Web-Anwendungen.** Da die Benutzerführung mit den Studenten über Web-Formulare realisiert ist, muss die Programmiersprache Mechanismen zur Generierung und Auswertung dieser Formulare bereitstellen.

1.3.4.2. Alternativen

Perl als Implementierungs-Sprache war zwar von Beginn an geplant, trotzdem wurde auch die Realisierung in anderen Programmiersprachen bzw. mit einer Kombination von verschiedenen Programmiersprachen erwogen.

Perl erfüllt die gestellten Anforderungen. Es gibt für jede der Anwendungen Module, die eine Realisierung erleichtern. Perl wird oft als „glue language“ bezeichnet, eine Sprache also, die sich Ideal für die Verknüpfung bereits bestehender Programme eignet. Im Vorwort von „Programming Perl“ heißt es

> Programming Perl von Larry Wall, Tom Christiansen und Randal L. Schwartz

> You want a language that makes it easy to manipulate numbers and text, files and directories, computers and networks, and especially programs. It should be easy to run external programs and scan their output for interesting tidbits. It should be easy to send those same tidbits off to other programs that can do special things with them. It should be easy to develop, modify, and debug your own programs too. And, of course, it should be easy to compile and run your programs, and do it portably, on any modern operating system. Perl does all that, and a whole lot more.

Perl bietet also alle Voraussetzungen zur Implementierung des Systems.

Als Alternativen wurden Python und Java in betracht gezogen. Python ist für ähnliche Aufgaben wie Perl geeignet, ist dabei aber im Unterschied zu Perl eine objekt-orientierte Sprache.

> Python Reference Manual von Guido van Rossum
>
> Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python ist also ebenso wie Perl für die Implementierung geeignet.

In Java wäre die Realisierung des ganzen Systems kaum möglich gewesen, da Java bei weitem nicht so gut für die Anbindung von externen Programmen geeignet ist wie Perl oder Python. Jedoch wäre eine teilweise Implementierung in Java, bspw. der Web-Schnittstelle mit Hilfe von Java-Servlets denkbar. Da aber sowohl die Web-Schnittstelle, als auch die Teile des Systems, die mit externen Programmen kommunizieren, teilweise die selben Funktionen benötigen, wäre eine teilweise doppelte Realisierung in zwei unterschiedlichen Programmiersprachen nicht vermeidbar.

In Anbetracht der aufgeführten Gründe, fiel die Entscheidung zugunsten von Perl aus. Die Entscheidung gegen Python und für Perl wurde von mir aufgrund meiner zu Beginn der Projektarbeit etwas größeren Erfahrung mit Perl und der mir zur Verfügung stehenden umfangreicheren Dokumentation zu Perl getroffen.

1.4. Implementierung des Systems

Im folgenden wird zunächst ein Überblick über das Gesamtsystem gegeben, und dann auf einzelne Komponenten näher eingangen.

1.4.1. Überblick

Das System läßt sich aus Benutzer-Sicht grob in drei Komponentengruppen aufteilen:

- Web-Schnittstelle zur Bedienung durch die Studenten
- Komponente zur Aufbereitung der durch die Studenten abgegebenen Lösungen für die Korrektur und Rückgabe der korrigierten Lösungen an die Studenten
- weitere organisatorische und administrative Hilfsmittel

Die Datenhaltung erfolgt vorrangig in einer SQL-Datenbank. Zusätzlich werden Daten, die regelmäßig von Hand verändert werden müssen (die Informationen/Beschreibungen zu den einzelnen Übungsblättern) in XML-Dateien verwaltet, welche bei Bedarf automatisch mit der Datenbank abgeglichen werden.

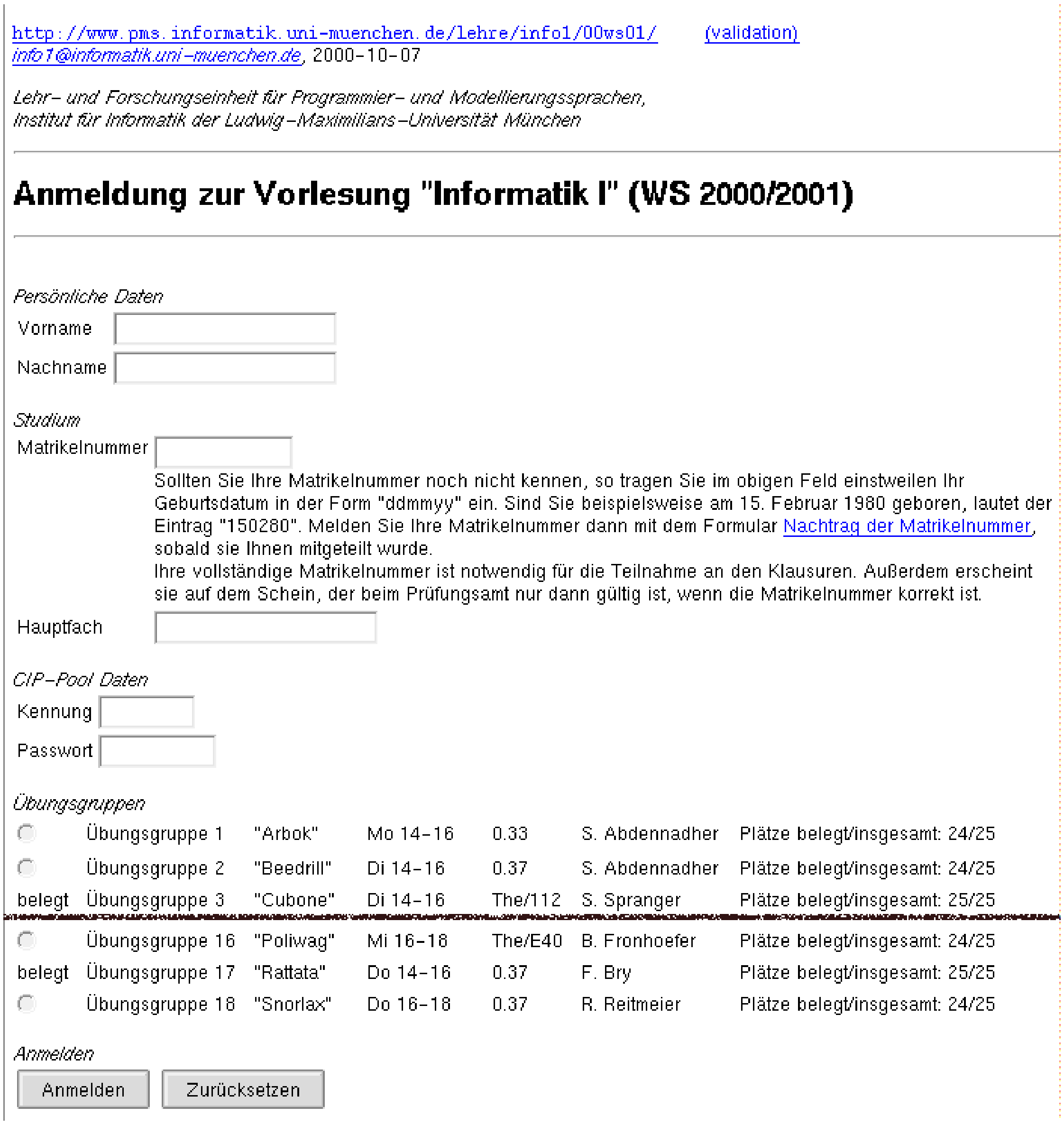
1.4.2. Web-Schnittstelle für die Studenten

Die Web-Schnittstelle besteht dabei im wesentlichen aus zwei HTML-Formularen. Eines dient zur Anmeldung der Studenten für den Übungsbetrieb, eines zur elektronischen Abgabe der Übungsaufgaben.

1.4.2.1. Anmeldung

Das Anmelde-Formular berücksichtigt die aktuelle Belegung der Übungsgruppen. Eine Anmeldung ist nur für Übungsgruppen mit noch freien Plätzen möglich. Abbildung 1-1 zeigt einen Screenshot des Anmelde-Formulars.

Abbildung 1-1. Anmelde-Formular für die Übungsgruppen



http://www.pms.informatik.uni-muenchen.de/lehre/info1/00ws01/ (validation)
info1@informatik.uni-muenchen.de, 2000-10-07

Lehr- und Forschungseinheit für Programmier- und Modellierungssprachen,
Institut für Informatik der Ludwig-Maximilians-Universität München

Anmeldung zur Vorlesung "Informatik I" (WS 2000/2001)

Persönliche Daten

Vorname

Nachname

Studium

Matrikelnummer

Sollten Sie Ihre Matrikelnummer noch nicht kennen, so tragen Sie im obigen Feld einstweilen Ihr Geburtsdatum in der Form "ddmmyy" ein. Sind Sie beispielsweise am 15. Februar 1980 geboren, lautet der Eintrag "150280". Melden Sie Ihre Matrikelnummer dann mit dem Formular Nachtrag der Matrikelnummer, sobald sie Ihnen mitgeteilt wurde.
Ihre vollständige Matrikelnummer ist notwendig für die Teilnahme an den Klausuren. Außerdem erscheint sie auf dem Schein, der beim Prüfungsamt nur dann gültig ist, wenn die Matrikelnummer korrekt ist.

Hauptfach

CIP-Pool Daten

Kennung

Passwort

Übungsgruppen

	Übungsgruppe 1	"Arbok"	Mo 14-16	0.33	S. Abdennadher	Plätze belegt/insgesamt: 24/25
	Übungsgruppe 2	"Beedrill"	Di 14-16	0.37	S. Abdennadher	Plätze belegt/insgesamt: 24/25
belegt	Übungsgruppe 3	"Cubone"	Di 14-16	The/112	S. Spranger	Plätze belegt/insgesamt: 25/25
	Übungsgruppe 16	"Poliwag"	Mi 16-18	The/E40	B. Fronhoefer	Plätze belegt/insgesamt: 24/25
belegt	Übungsgruppe 17	"Rattata"	Do 14-16	0.37	F. Bry	Plätze belegt/insgesamt: 25/25
	Übungsgruppe 18	"Snorlax"	Do 16-18	0.37	R. Reitmeier	Plätze belegt/insgesamt: 24/25

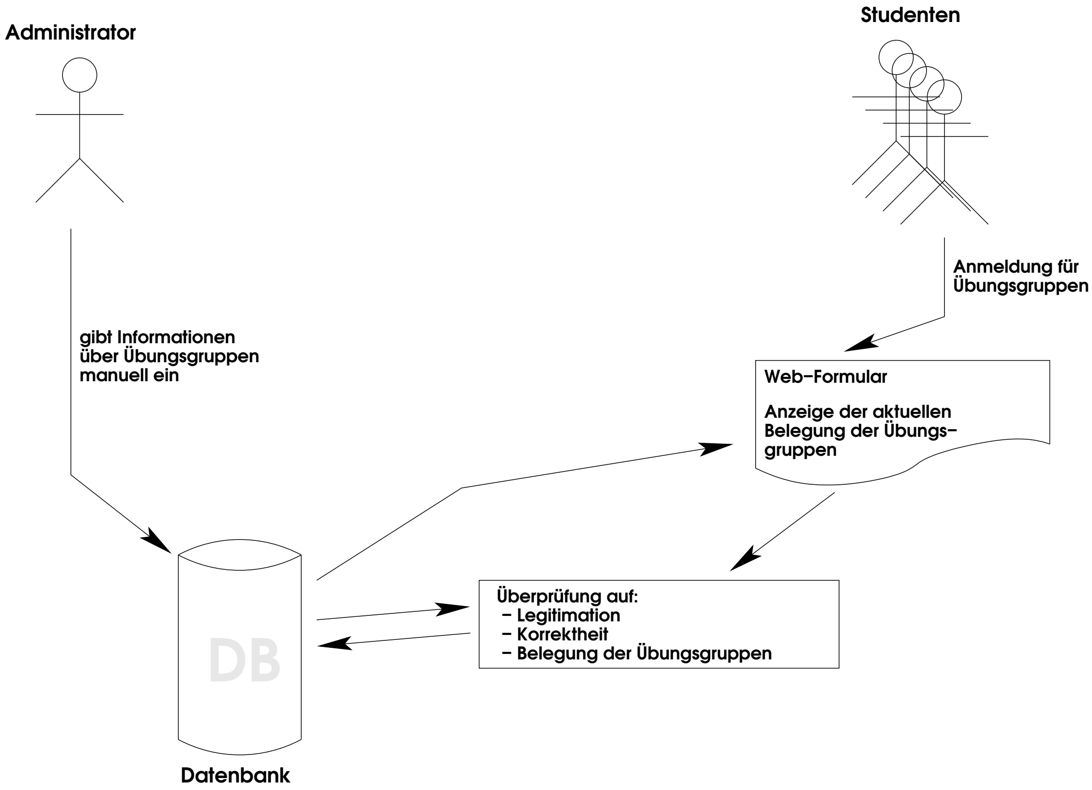
Anmelden

Anmelden Zurücksetzen

Anmelde-Formular für die Übungsgruppen

Die Anzahl der insgesamt zur Verfügung stehenden Plätze wird vom Administrator in die Datenbank eingetragen. Die Anzahl der bereits belegten Plätze wird dynamisch beim Aufruf der Webseite aufgrund der bereits erfolgten Anmeldungen bestimmt. Für die Übungsgruppen anmelden können sich nur Studenten mit gültiger Kennung am CIP-Pool des Instituts für Informatik. Abbildung 1-2 stellt den Anmeldevorgang grafisch dar.

Abbildung 1-2. Anmeldung der Studenten für die Übungsgruppen



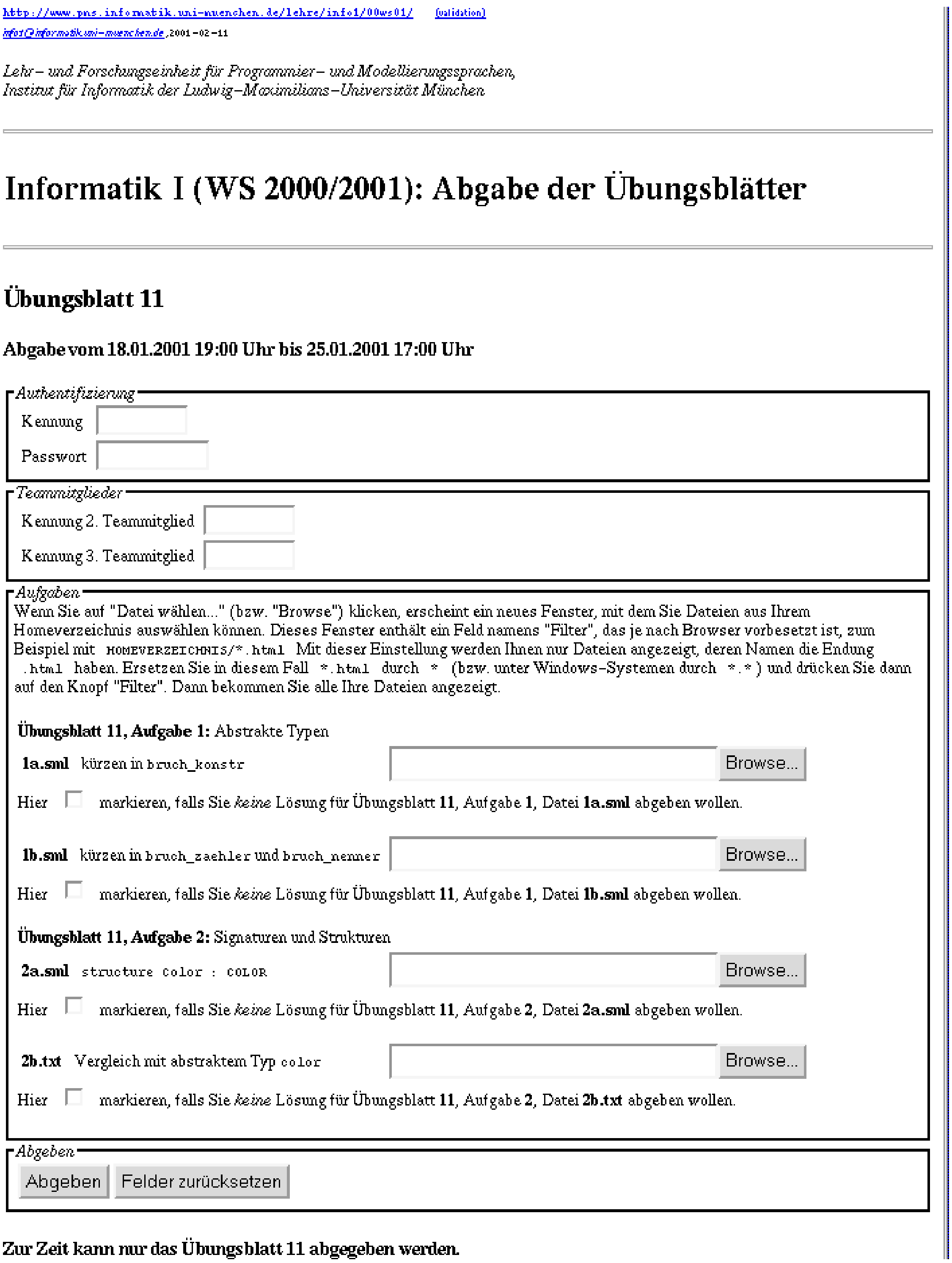
Die Studenten bekommen bei einem Fehler bei der Anmeldung eine Web-Seite mit einer entsprechenden Meldung angezeigt. Im Erfolgs-Fall wird zusätzlich zu einer Meldung im Browser auch eine E-Mail an die Studenten geschickt, welche nochmals die eingegebenen Daten beinhaltet und einen Hinweis gibt, wie man die Daten nachträglich korrigieren kann.

1.4.2.2. Abgabe der Übungsblätter

Es wird immer das aktuell gültige Abgabe-Formular angezeigt. Können zu einem Zeitpunkt mehrere Übungsblätter abgegeben werden, so werden am Ende des Formulars Verweise auf die Abgabe-Formulare für die anderen Übungsblätter angezeigt.

Das Abgabe-Formular (Abbildung 1-1) wird dynamisch aus einer XML-Konfigurationsdatei und aus einem HTML-Template erzeugt. In der Konfigurationsdatei werden u.a. die abzugebenden Aufgaben angegeben.

Abbildung 1-3. Abgabe-Formular für die Übungsblätter


http://www.pms.informatik.uni-muenchen.de/lehre/info1/00ws01/ (validation)
info1@informatik.uni-muenchen.de, 2001-02-11

Lehr- und Forschungseinheit für Programmier- und Modellierungssprachen,
Institut für Informatik der Ludwig-Maximilians-Universität München

Informatik I (WS 2000/2001): Abgabe der Übungsblätter

Übungsblatt 11

Abgabe vom 18.01.2001 19:00 Uhr bis 25.01.2001 17:00 Uhr

Authentifizierung

Kennung

Passwort

Teammitglieder

Kennung 2. Teammitglied

Kennung 3. Teammitglied

Aufgaben

Wenn Sie auf "Datei wählen..." (bzw. "Browse") klicken, erscheint ein neues Fenster, mit dem Sie Dateien aus Ihrem Homeverzeichnis auswählen können. Dieses Fenster enthält ein Feld namens "Filter", das je nach Browser vorbesetzt ist, zum Beispiel mit `HOMEVERZEICHNIS/*.html` Mit dieser Einstellung werden Ihnen nur Dateien angezeigt, deren Namen die Endung `.html` haben. Ersetzen Sie in diesem Fall `*.html` durch `*` (bzw. unter Windows-Systemen durch `*.*`) und drücken Sie dann auf den Knopf "Filter". Dann bekommen Sie alle Ihre Dateien angezeigt.

Übungsblatt 11, Aufgabe 1: Abstrakte Typen

1a.sml kürzen in `bruch_konstr` Browse...

Hier markieren, falls Sie *keine* Lösung für Übungsblatt **11**, Aufgabe **1**, Datei **1a.sml** abgeben wollen.

1b.sml kürzen in `bruch_zaehler` und `bruch_nenner` Browse...

Hier markieren, falls Sie *keine* Lösung für Übungsblatt **11**, Aufgabe **1**, Datei **1b.sml** abgeben wollen.

Übungsblatt 11, Aufgabe 2: Signaturen und Strukturen

2a.sml `structure color : COLOR` Browse...

Hier markieren, falls Sie *keine* Lösung für Übungsblatt **11**, Aufgabe **2**, Datei **2a.sml** abgeben wollen.

2b.txt Vergleich mit abstraktem Typ `color` Browse...

Hier markieren, falls Sie *keine* Lösung für Übungsblatt **11**, Aufgabe **2**, Datei **2b.txt** abgeben wollen.

Abgeben

Abgeben | Felder zurücksetzen

Zur Zeit kann nur das Übungsblatt 11 abgegeben werden.

Abgabe-Formular für die Übungsblätter

Die erfolgreiche Abgabe ist nur bei Erfüllung folgender Punkte möglich:

- alle Team-Mitglieder sind in einer Übungsgruppe angemeldet
- alle Team-Mitglieder werden vom selben Korrektor betreut
- keines der Team-Mitglieder hat für dieses Übungsblatt bereits eine Lösung abgegeben
- es wurde mindestens eine Aufgabe bearbeitet
- die Dateien sind alle im ASCII-Format
- die Programmieraufgaben sind syntaktisch korrekt

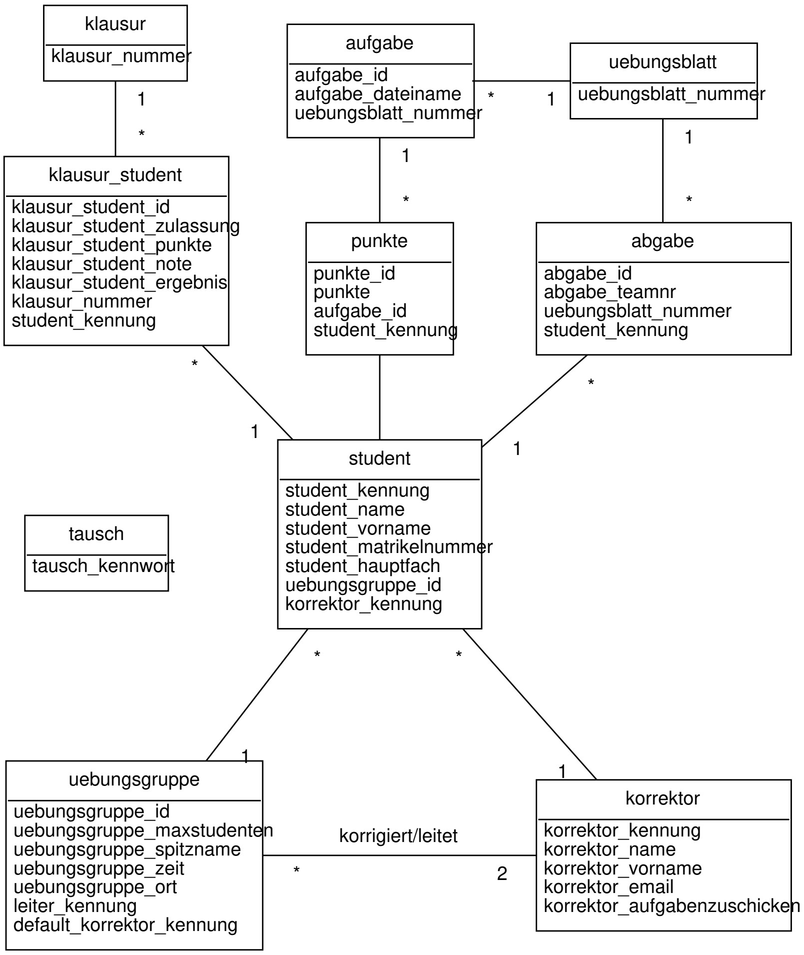
Je nach (Nicht-)Erfüllung der obigen Punkte erscheint eine entsprechende Meldung. Bei erfolgreicher Abgabe wird, ebenso wie bei der Anmeldung, zusätzlich auch eine E-Mail an alle Team-Mitglieder verschickt.

1.4.3. Datenhaltung

Die für den Übungsbetrieb relevanten Informationen werden in der Datenbank verwaltet. Dies umfasst die Informationen zu den Studenten, wie Vor- und Nachname, Matrikelnummer, die erreichten Punktzahlen für die einzelnen Aufgaben und die Punkte in der Klausur. Weiterhin werden in der Datenbank die Zuordnungen zwischen Korrektor und Student, Übungsgruppen-Leiter und Übungsgruppe sowie Student und Übungsgruppe abgespeichert. Abbildung 1-4 gibt das Schema der

Datenbank wieder.

Abbildung 1-4. Datenbank-Schema

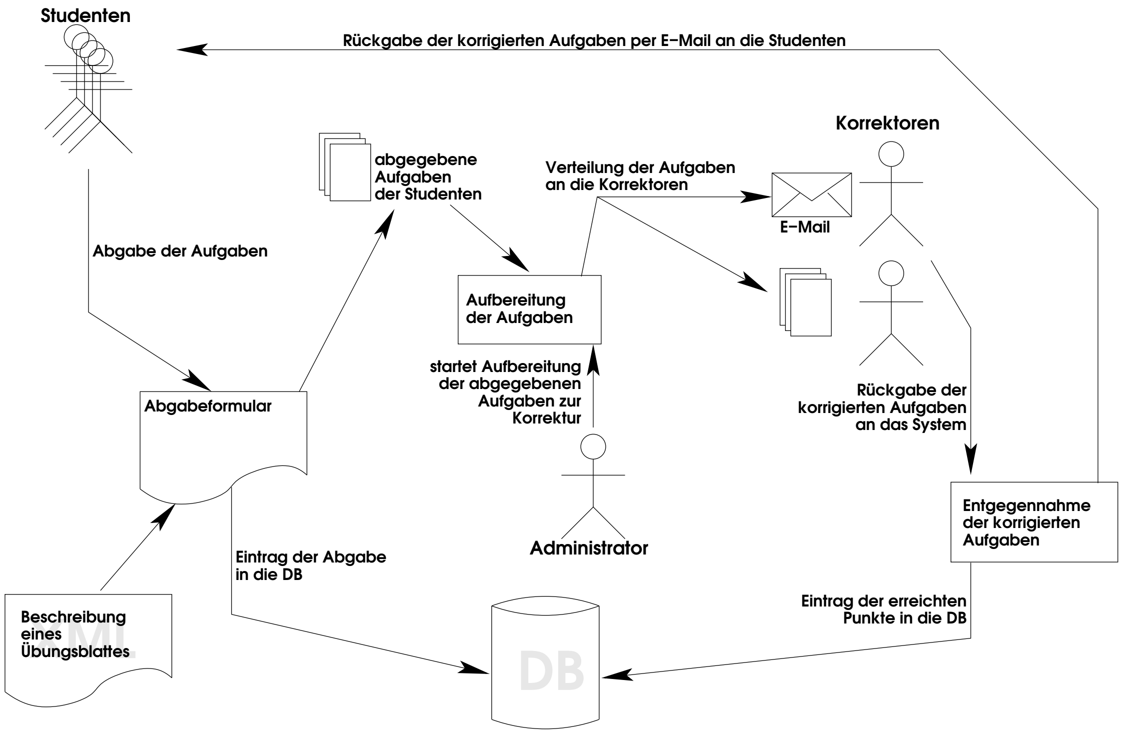


In der Tabelle korrektor werden sowohl die Korrektoren als auch die Übungsgruppen-Leiter gespeichert. Daher sind jeder Übungsgruppe zwei Records aus der Tabelle korrektor zugeordnet.

1.4.4. Elektronisch unterstützte Korrektur der Aufgaben

Die von den Studenten elektronisch abgegebenen Aufgaben werden auch mit elektronischer Hilfe korrigiert. Dazu werden sie mit für die Korrektur hilfreichen Informationen versehen. Anschließend werden die Dateien an die Korrektoren weitergeleitet. Nach erfolgter Korrektur geben die Korrektoren die Dateien wieder elektronisch ab. Diese Abgabe erfolgt dabei entweder über den Aufruf eines Skripts auf dem Server, oder durch Verwendung eines Web-Formulars. In beiden Fällen werden die Punktzahlen aus den korrigierten Aufgaben ausgelesen und in die Datenbank übertragen sowie die korrigierten Aufgaben an die Studenten verschickt. Die jeweils betroffenen Korrektoren und Übungsgruppen-Leiter erhalten eine aktuelle Punkteliste per E-Mail zugeschickt. Abbildung 1-5 stellt den Ablauf der elektronisch unterstützten Korrektur dar.

Abbildung 1-5. Ablauf der elektronisch unterstützten Korrektur



1.4.4.1. Automatische Verarbeitung der Lösungen

Die von den Studenten abgegebenen Lösungen werden zuerst in einem Verzeichnis abgelegt. Sobald der Abgabe-Zeitraum für ein Übungsblatt zu Ende ist, startet der Administrator des Systems die weitere Verarbeitung der Lösungen.

- Archivierung der abgegebenen Lösungen
- Vergleich der Lösungen/Plagiatssuche
- Test der Lösungen
- Einfügen von Zusatzinformationen in die Dateien wie
  - Übungsblatt
  - Aufgabe und Thema

- maximal erreichbare Punktzahl
- Name und Kennung des Studenten
- Namen und Kennungen der Team-Mitglieder

- Verteilung der Lösungen an die Korrektoren

Die Ergebnisse der Plagiatssuche und der automatischen Tests werden dabei direkt in die jeweiligen Dateien als Kommentar eingefügt.

Den Korrektoren werden die von ihnen zu korrigierenden Lösungen in ein Verzeichnis kopiert und auf Wunsch auch per E-Mail zugeschickt.

1.4.4.2. Korrektur der Lösungen

Den Korrektoren stehen jetzt die von den Studenten abgegebenen Lösungen in elektronischer Form zur Verfügung. Zusätzlich ist jede Datei mit Informationen angereichert, die die Korrektur erleichtern.

Beispiel 1-1. Lösung einer Programmieraufgabe mit Zusatzinformationen

```
(*
INFO-START

Übungsblatt: 05
Aufgabe: 3
Thema: iterative Prozesse
Dateiname: 3.sml
erreichbare Punktzahl: 8
erreichte Punktzahl: 0

Team: Sybille Schielenski (schielen)
Korrektor: G. Schneemayer

INFO-ENDE
*)

fun quadrat(x)     = x * x;
fun ist_gerade(n) = (n mod 2) = 0;
fun haelfte(n)     = n div 2;

[...]

(*
VERGLEICH-START
```

```
Diese Datei ist nach Stufe 2 gleich mit den Lösungen der Teams\
  (insgesamt 1): laqtib
VERGLEICH-ENDE
*)

(*
TEST-START

Die folgenden Tests sind fehlgeschlagen:

potenz_iter
 Ergebnis: Error: unbound variable or constructor: potenz_iter\
          <=== Diskrepanz
 Erwartet: fn : int * int -> int
potenz_iter(0,0)
 Ergebnis: Error: unbound variable or constructor: potenz_iter\
          <=== Diskrepanz
 Erwartet: 1 : int
[...]
TEST-ENDE
*)

```

Korrigiert wird direkt in den von den Studenten abgegebene Dateien. Der Korrektor muss dabei mindestens die erreichte Punktzahl angeben, kann aber seinerseits beliebige Kommentare in die Datei einfügen, solange der INFO-Kommentar bestehen bleibt.

Nach erfolgter Korrektur starten die Korrektoren ihrerseits wieder ein Skript, welches die korrigierten Lösungen an die Studenten verschickt und die erreichten Punkte in die Datenbank einträgt. Dabei wird überprüft, ob jede Lösung korrigiert wurde und die Punktzahl nicht größer als die maximal erreichbare Punktzahl ist.

Die Korrektoren und die Übungsgruppen-Leiter bekommen im Erfolgs-Fall eine Liste mit den aktuellen Punktzahlen ihrer Übungsgruppen zugeschickt.

Kapitel 2. Bedienungsanleitung zur Korrektur

In diesem Teil wird das Vorgehen bei der Korrektur der Übungsaufgaben unter Verwendung des elektronischen Abgabesystems beschrieben. Die konkreten Bezeichner beziehen sich dabei auf die Installation des Systems für "Informatik I" im Wintersemester 2000/2001.

2.1. Was vor der Korrektur passiert

Die Studenten geben Ihre Übungsblätter in Teams mit bis zu drei Personen über ein HTML-Formular ab. Die Abgabe der Aufgaben ist nur zu den dafür vorgesehenen Zeiten möglich. Bei der Abgabe werden u.a. folgende Punkte sichergestellt:

- keine mehrfache Abgabe durch einzelne Teammitglieder
- alle Teammitglieder sind beim gleichen Korrektor
- Dateiformat ist „text“
- SML Programme sind syntaktisch korrekt

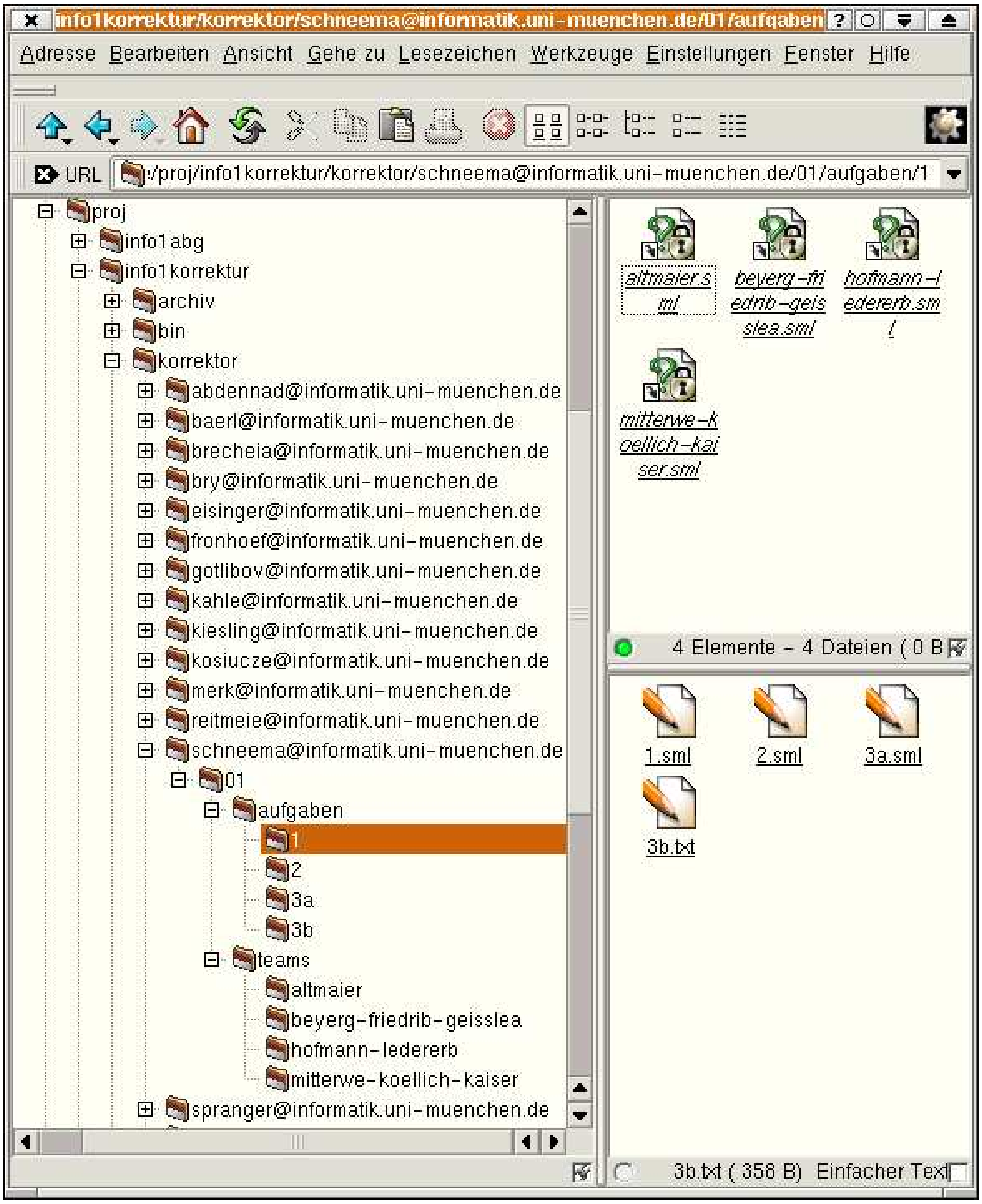
Bei erfolgreicher Abgabe erhalten alle Teammitglieder eine E-Mail als Bestätigung.

Nach Ende des Abgabezeitraums werden die abgegebenen Aufgaben den Korrektoren in einem Verzeichnis und optional zusätzlich per E-Mail zugänglich gemacht. Die Aufgaben werden dabei um einen Kommentar am Anfang der Datei ergänzt.

2.1.1. Verzeichnisstruktur

Jeder Korrektor hat auf dem Server für das Abgabesystem ein eigenes Verzeichnis mit dem Namen seiner E-Mail Adresse (unter `/home/proj/info1korrektur/korrektor/`). Für jedes Übungsblatt wird hierunter ein Verzeichnis mit der Nummer des Übungsblatts angelegt. Darunter werden die abgegebenen Lösungen im Verzeichnis `teams` nach Teams sortiert und mit Hilfe symbolischer Links im Verzeichnis `aufgaben` nach Aufgaben sortiert abgelegt. Abbildung 2-1 zeigt in einem Screenshot die Verzeichnisstruktur.

Abbildung 2-1. Verzeichnisstruktur zur Korrektur



Sortierung nach Aufgaben und Teams. Links die Verzeichnisstruktur, rechts oben Datei `1.sml` sortiert nach Teams und rechts unten Team „altmaier“.

2.1.2. Zusatzinformationen

Die Aufgaben werden mit Zusatzinformationen u.a. über das Team und die maximal erreichbare Punktzahl versehen. Das Beispiel zeigt die Aufgabe `1.sml` von Übungsblatt 1, abgegeben von „Simone Mitterweger“ mit der Kennung „mitterwe“. Ihre Teammitglieder sind „Florian Köllich“ und „Christoph Kaiser“, Korrektor ist „G. Schneemayer“. Die Zusatzinformationen sind immer Kommentare zu Beginn der Datei, sie beginnen mit *INFO-START* und enden mit *INFO-ENDE*.

Beispiel 2-1. Datei `1.sml` mit Zusatzinformationen

```
(*
INFO-START

Übungsblatt: 01
Aufgabe: 1
Thema: Syntaxfehler
Dateiname: 1.sml
erreichbare Punktzahl: 1
erreichte Punktzahl:

Team: Simone Mitterweger (mitterwe), Florian Köllich (koellich),\
  Christoph KAISER (kaiser)
Korrektor: G. Schneemayer

INFO-ENDE
*)
fun quadrat(x)           = x*x;
fun summe_quadrate(x,y)  = quadrat(x)*quadrat(x);
fun pythagoras(a,b,c)    = (summe_quadrate(a,b) = quadrat(c));
fun pythagoraszahl(z)    = pythagoras(z-1, z, z+1);
val ist_4_pythagoraszahl = pythagoraszahl(4);
```

2.2. Korrektur

Im folgenden wird das Vorgehen bei der Korrektur beschrieben. Zuerst bei Zugriff auf das Korrektoren-Verzeichnis auf dem Server (`/home/proj/info1korrektur/korrektor/`), danach von beliebigen Rechner über E-Mail und HTML-Formular. Da beide Vorgehensweisen in weiten Teilen ähnlich sind, werden im zweiten Teil nur kurz die Unterschiede aufgezeigt.

2.2.1. Korrektur bei Zugriff auf das Korrektoren-Verzeichnis (`/home/proj/info1korrektur/korrektor/`)

Von einem Rechner mit Zugriff auf das Korrektoren-Verzeichnis (`/home/proj/info1korrektur/korrektor/`), kann die Korrektur direkt in diesem Verzeichnis erfolgen.

Jeder Korrektor hat Schreibrechte in diesem Verzeichnis. Die Korrektur erfolgt direkt in den Dateien unterhalb dieses Verzeichnisses. Dabei muss die erreichte Punktzahl in den Zusatzinformationen zu den einzelnen Dateien eingetragen werden. Auch längere Kommentare müssen direkt in diese Dateien eingetragen werden, es dürfen in diesem Verzeichnis *keine* zusätzlichen Dateien angelegt werden.

> **Wichtig:** Es dürfen nur ganze Punkte vergeben werden. Die zu vergebende Punktzahl für die Aufgaben ist also entsprechend groß zu wählen, so dass eine genügend feine Abstufung möglich ist.

Da die Sortierung nach Aufgaben durch symbolische Links die auf die eigentlichen Dateien in den Team-Verzeichnissen zeigen erfolgt, ist es gleichgültig, welche Sortierung man zur Korrektur benutzt, da immer die Dateien in den Team-Verzeichnissen bearbeitet werden.

Der Bereich zwischen INFO-START und INFO-ENDE sollte außer dem Eintrag der Punktzahl nicht geändert werden. Außerhalb dieses Bereichs können beliebige Kommentare hinzugefügt werden. Es ist jedoch sinnvoll, Kommentare zu der Lösung eindeutig als Kommentare des Korrektors zu Kennzeichnen.

Beispiel 2-2. korrigierte Datei `1.sml`

```
(*
INFO-START

Übungsblatt: 01
Aufgabe: 1
Thema: Syntaxfehler
```

```
Dateiname: 1.sml
erreichbare Punktzahl: 1
erreichte Punktzahl: 1

Team: Simone Mitterweger (mitterwe), Florian Köllich (koellich),\
  Christoph KAISER (kaiser)
Korrektor: G. Schneemayer

INFO-ENDE
*)
(*g
Zusätzliche Kommentare können
g*)
fun quadrat(x)           = x*x; (*g überall in g*)
fun summe_quadrate(x,y)  = quadrat(x)*quadrat(x);
fun pythagoras(a,b,c)    = (summe_quadrate(a,b) = quadrat(c));
fun pythagoraszahl(z)    = pythagoras(z-1, z, z+1);
(*g in der Datei stehen g*)
val ist_4_pythagoraszahl = pythagoraszahl(4);
```

Nachdem alle Dateien korrigiert worden sind, muss nun einfach das bash-Skript **korrigiert.sh** auf dem Server (info1.informatik.uni-muenchen.de) im jeweiligen Übungsblatt-Verzeichnis aufgerufen werden. Da der Zugriff auf die Datenbank nur vom Server erlaubt ist, funktioniert dieses Skript nur, wenn es auf dem Server (beispielsweise per Login mit ssh) ausgeführt wird.

korrigiert.sh überprüft ob bei allen Aufgaben die erreichte Punktzahl eingetragen wurde und ob sie unter der maximal erreichbaren Punktzahl liegt. Stößt das Skript auf eine Datei für die diese Bedingung nicht erfüllt ist, bricht es mit einer Fehlermeldung mit Angabe der betroffenen Datei ab. Das Skript erkennt also automatisch, wenn bei der Korrektur einzelne Aufgaben vergessen wurden. Die Rückgabe einer unvollständigen Korrektur ist nicht möglich.

Sind alle Aufgaben korrigiert, trägt das Programm die erreichten Punktzahlen in die Datenbank ein und verschickt an die Teams die korrigierten Aufgaben per E-Mail. Der Korrektor und der Leiter der jeweiligen Übungsgruppe erhält eine E-Mail mit den aktuellen Punktzahlen der Studenten. Sind für einen Studenten bereits Punkte in der Datenbank, dann gibt das Skript eine entsprechende Meldung aus, die weitere Verearbeitung geht jedoch normal weiter.

Damit ist die Korrektur abgeschlossen.

2.2.2. Korrektur auf beliebigem Rechner

Ist der Zugang auf das Korrektoren-Verzeichnis auf dem Server möglich, so kann man sich zusätzlich die Aufgaben per E-Mail als Anhang zuschicken lassen.

Wird das Attachment mit **tar xfz Dateiname** entpackt entsteht die gleiche Struktur wie im Verzeichnis `/home/proj/info1korrektur/korrektor/korrektor_e-mail/`.

Die Korrektur erfolgt dann ebenso wie bei der Korrektur im Verzeichnis `/home/proj/info1korrektur/korrektor/korrektor_e-mail/`.

Nach erfolgter Korrektur müssen die korrigierten Aufgaben wieder gepackt werden. Dies geschieht mit **tar czf Archivname Verzeichnisname**. Alternativ kann auch ein zip-Archiv (z.B. mit WinZip) erstellt und zurückgegeben werden. In beiden Fällen ist darauf zu achten, dass die Verzeichnisstruktur die gleiche ist, wie bei dem per E-Mail zugeschickten Archiv. Das Archiv muss beim entpacken ein Verzeichnis mit der Nummer des Übungsblattes (z.B. 07 für das 7. Übungsblatt) anlegen, darunter ein Verzeichnis `teams`. Das Unterverzeichnis `aufgaben` muss nicht vorhanden sein.

Die gepackte Datei kann über das Formular zur Abgabe der korrigierten Übungsaufgaben (https://info1.informatik.uni-muenchen.de/korrekturformular.html) unter Angabe der CIP-Pool Kennung und des CIP-Pool Passworts abgegeben werden. Für diese Abgabe gilt das gleiche wie für das Skript **korrigiert.sh**. Sind noch nicht alle Aufgaben korrigiert oder sind mehr als die maximale Punktzahl vergeben worden, wird die Annahme verweigert. Im Erfolgsfall werden ebenso wie bei **korrigiert.sh** E-Mails an die Studenten mit den korrigierten Aufgaben geschickt und die Punkte in die Datenbank eingetragen. Sowohl der Korrektor und der jeweilige Übungsgruppen-Leiter bekommen als Bestätigung per E-Mail den aktuellen Punktestand zugeschickt.

Beispiel 2-3. Korrektur Übungsblatt 01

Das Vorgehen bei Übungsblatt 01 ist wie folgt:

1. Attachment speichern unter dem Namen `01.tgz`
2. Archiv entpacken mit **tar xfz 01.tgz**
3. im entstandenen Verzeichnis `01` korrigieren
4. Verzeichnis `01` mit **tar czf 01.tgz 01** packen
5. Datei unter der URL https://info1.informatik.uni-muenchen.de/korrekturformular.html abgeben
   - bei Meldung über fehlende oder falsche Punkteinträge zurück zu Punkt 3
   - bei Erfolgsmeldung: fertig!

Kapitel 3. Bedienung Administrator

Die Bedienung durch den Administrator wird in diesem Teil anhand der typischerweise durchzuführenden Aufgaben gezeigt. Auf die Optionen der einzelnen Programme wird hier nicht näher eingegangen. Hinweise dazu finden sich in Anhang A.

3.1. Konfiguration zu Semesterbeginn

Nach erfolgreicher Installation des Systems für eine Veranstaltung (siehe Kapitel 4) müssen noch einige Einstellungen zum Ablauf und zur Organisation am System vorgenommen werden. Einige dieser Einstellungen können sich auch noch während des Semesters ändern.

3.1.1. Eintragungen in der Datanbank

Die meisten dieser Einstellungen werden in der Datenbank gemacht. In ihr müssen zum Betrieb des Systems einige Daten über die Organisation der Veranstaltung gespeichert sein.

3.1.1.1. Korrektoren und Übunsgruppen-Leiter

Die Korrektoren und Übungsgruppen-Leiter werden dem System durch Einträge in der Tabelle korrektor bekannt gemacht.

In dieser Tabelle müssen die Korrektoren *und* die Übungsgruppen-Leiter eingetragen werden. Die meisten Felder dieser Tabelle sind selbsterklärend, einige Dinge sind jedoch besonders zu beachten:

korrektor_kennung

CIP-Pool Kennung des Korrektors. Jeder Korrektor benötigt eine Kennung am CIP-Pool.

korrektor_aufgabenzuschicken

1, wenn der Korrektor sich die von ihm zu korrigierenden Lösungen auch per E-Mail zuschicken lassen will

0, sonst

korrektor_email

E-Mail des jeweiligen Korrektors bzw. Übungsgruppen-Leiters. Die E-Mail Adresse ist unbedingt erforderlich, auch wenn die Lösungen nicht per E-Mail zugeschickt werden sollen, so erhält der Korrektor bzw. der Übungsgruppen-Leiter doch die aktuellen Punktestände per E-Mail zugeschickt.

> Auch das Unterverzeichnis im Korrektoren-Verzeichnis hat als Namen die E-Mail Adresse des Korrektors

Die Datenbank wird bspw. mit folgenden Befehlen gefüllt:

```
INSERT INTO korrektor\
  (korrektor_name, korrektor_vorname, korrektor_email,\
  korrektor_aufgabenzuschicken, korrektor_kennung)\
  VALUES ('Abdennadher','Slim','abdennad@informatik.uni-muenchen.de',\
  1,'abdennad');
INSERT INTO korrektor\
  (korrektor_name, korrektor_vorname, korrektor_email,\
  korrektor_aufgabenzuschicken, korrektor_kennung)\
  VALUES ('Bry','Francois','bry@informatik.uni-muenchen.de',\
  0,'bry');
INSERT INTO korrektor\
  (korrektor_name, korrektor_vorname, korrektor_email,\
  korrektor_aufgabenzuschicken, korrektor_kennung)\
  VALUES ('Eisinger','Norbert','eisinger@informatik.uni-muenchen.de',\
  1,'eisinger');
INSERT INTO korrektor\
  (korrektor_name, korrektor_vorname, korrektor_email,\
  korrektor_aufgabenzuschicken, korrektor_kennung)\
  VALUES ('Fronhoefer','Bertram','fronhoef@pms.informatik.uni-muenchen.de',\
  1,'fronhoef');
[...]
```

3.1.1.2. Übungsgruppen

Ebenso wie die Korrektoren und Übungsgruppen-Leiter müssen auch die Übungsgruppen selbst dem System bekannt gemacht werden.

Die Studenten melden sich zu Beginn des Semesters für eine Übungsgruppe an. Um zu vermeiden, dass einzelne Übungsgruppen zu voll werden, muss für jede Übungsgruppe eine Obergrenze an zugelassenen Teilnehmern eingetragen werden.

Das System verlangt von den Studenten, dass, falls sie im Team abgeben, alle Team-Mitglieder den selben Korrektor haben. Um den Studenten ihre Zuordnung zu einzelnen Korrektoren durchschaubar zu machen, ist es sinnvoll, dass genau ein Korrektor einer Übungsgruppe zugeordnet ist.

Die Felder in der Tabelle uebungsgruppe sind auch wieder selbsterklärend, zu beachten ist nur folgendes:

uebungsgruppe_id

Der Schlüssel der Tabelle. Er wird von MySQL automatisch belegt. Zu beachten ist, dass die Übungsgruppen in der Reihenfolge ihres Einfügens in der Datenbank ihre Nummern (eben den Schlüssel) zugewiesen bekommen. Entweder müssen die Übungsgruppen also in der gewünschten Reihenfolge (sinnvoll wäre eine zeitliche Anordnung) in die Datenbank eingetragen werden, oder der Schlüssel-Wert muss manuell gesetzt werden um eine sinnvolle Nummerierung der Übungsgruppen zu erreichen.

default_korrektor_kennung

Die Kennung des Korrektors, dem die Studenten dieser Übungsgruppe automatisch zugewiesen werden. Die Zuordnung der Studenten zu den einzelnen Korrektoren kann auch nachträglich noch geändert werden (Feld korrektor_kennung, Tabelle student), dabei ist jedoch auf die oben besprochene Problematik bei der Abgabe im Team zu achten.

Die Tabelle uebungsgruppe kann mit folgenden SQL-Anweisungen gefüllt werden:

```
INSERT INTO uebungsgruppe\
  (uebungsgruppe_maxstudenten, leiter_kennung, default_korrektor_kennung,\
  uebungsgruppe_spitzname, uebungsgruppe_zeit, uebungsgruppe_ort)\
  VALUES (25,'abdennad','baerl','Arbok','Mo 14-16','0.33');
INSERT INTO uebungsgruppe\
  (uebungsgruppe_maxstudenten, leiter_kennung, default_korrektor_kennung,\
  uebungsgruppe_spitzname, uebungsgruppe_zeit, uebungsgruppe_ort)\
  VALUES (25,'abdennad','abdennad','Beedrill','Di 14-16','0.37');
INSERT INTO uebungsgruppe\
  (uebungsgruppe_maxstudenten, leiter_kennung, default_korrektor_kennung,\
  uebungsgruppe_spitzname, uebungsgruppe_zeit, uebungsgruppe_ort)\
  VALUES (25,'spranger','spranger','Cubone','Di 14-16','The/112');
[...]
```

3.1.1.2.1. Wechsel von Übungsgruppen

Grundsätzlich ist der Wechsel von Übungsgruppen nicht möglich. Den Studenten soll jedoch nach vorheriger Genehmigung die Möglichkeit gegeben werden, Übungsgruppen zu tauschen.

Um diese Genehmigung durchführen zu können, existiert die Tabelle tausch. In dieser Tabelle sind Tausch-Kennwörter eingetragen, jedes dieser Wörter kann dabei genau einmal von zwei Studenten zum Tausch ihrer Übungsgruppen über das Tausch-Formular (siehe Abschnitt A.1.2.4) benutzt werden.

Beispiel für das Füllen der Tabelle tausch:

```
INSERT INTO tausch (tausch_kennwort) VALUES ('ApR/f2Qn');
INSERT INTO tausch (tausch_kennwort) VALUES ('7&D3e6q>5');
INSERT INTO tausch (tausch_kennwort) VALUES ('?nWt=k7AC');
```

```
[...]
```

3.1.1.3. Klausuren

Da die Ergebnisse der Klausuren auch in der Datenbank gespeichert werden, muss das System auch über die Anzahl der Klausuren bescheid wissen. In der Tabelle klausur ist dazu für jede der Klausuren ein eigener Eintrag nötig:

```
INSERT INTO klausur (klausur_nummer) VALUES (1);
INSERT INTO klausur (klausur_nummer) VALUES (2);
```

3.1.1.4. Übungsblätter

Analog zu den Klausuren müssen auch die Übungsblätter in die Datenbank eingetragen werden:

```
INSERT INTO uebungsblatt (uebungsblatt_nummer) VALUES (1);
INSERT INTO uebungsblatt (uebungsblatt_nummer) VALUES (2);
INSERT INTO uebungsblatt (uebungsblatt_nummer) VALUES (3);
[...]
```

3.1.2. XML-Konfigurationsdateien

Die Informationen zu den einzelnen Übungsblättern wird Aufgrund der leichteren Editierbarkeit nicht in der Datenbank sondern in XML-Konfigurationsdateien gespeichert. Die Konfigurationsdateien zu den einzelnen Übungsblättern werden in Abschnitt 3.2.1.1 besprochen.

Die Festlegung der Abgabezeiträume für die Übungsblätter finden jedoch im Allgemeinen bereits zu Beginn des Semesters statt, und so soll hier auf die Konfiguration der Datei `uebungsblaetter.xml` hingewiesen werden. Näheres zur Syntax findet sich in Abschnitt A.1.1.2.

3.2. Konfiguration für ein Übungsblatt

Dieser Abschnitt beschreibt beispielhaft die Konfiguration für das 10. Übungsblatt der Vorlesung "Informatik I" im Wintersemester 2000/2001.

3.2.1. Das Übungsblatt 10

Bevor auf die Konfiguration des Übungsblatts eingegangen werden kann, muss das Übungsblatt noch kurz vorgestellt werden.

Das Übungsblatt 10 besteht aus drei abzugebenden Aufgaben. Zwei der Aufgaben (Aufgabe 1 und 4, Aufgabe 3 ist nicht abzugeben) sind keine Programme sondern Text. Aufgabe 2 hingegen ist ein SML-Programm. Bei allen Aufgaben soll ein Plagiatstest durchgeführt werden, bei Aufgabe 1 in der 2. Stufe, bei den anderen beiden in der 1. Stufe. Für die Programmieraufgabe (Aufgabe 2) muss sowohl zum Testen der Syntax als auch zur Durchführung der automatischen Tests eine andere Datei zusätzlich geladen werden.

Nun zur Konfiguration für dieses Übungsblatt.

3.2.1.1. Beschreibung des Übungsblatts in `uebungsblatt10.xml`

Beispiel 3-1. Datei `conf/xml/uebungsblatt10.xml`

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>

<!DOCTYPE uebungsblatt [
   <!ELEMENT uebungsblatt    (nummer, information?, aufgaben) >
   <!ELEMENT nummer          (#PCDATA)                        >
   <!ELEMENT information     (#PCDATA)                        >
   <!ELEMENT aufgaben        (aufgabe)*                       >
   <!ELEMENT aufgabe         (nummer, thema, datei+)          >
   <!ELEMENT thema           (#PCDATA)                        >
   <!ELEMENT datei           (name, punkte, beschreibung,
                              vergleich?)                     >
   <!ELEMENT name            (#PCDATA)                        >
   <!ELEMENT punkte          (#PCDATA)                        >
   <!ELEMENT beschreibung    (#PCDATA)                        >
   <!ELEMENT vergleich       (#PCDATA)                        >
]>

<uebungsblatt>
  <nummer>10</nummer>

  <aufgaben>

    <aufgabe>
      <nummer>1</nummer>
      <thema><![CDATA[statische/dynamische Bindung]]&gt;</thema>
      <datei>
        <name>1.txt</name> <punkte>12</punkte>
        <beschreibung>
          <![CDATA[Teilaufgaben a) b) c)]]&gt;
        </beschreibung>
```

```
      <vergleich>2</vergleich>
    </datei>
  </aufgabe>

  <aufgabe>
    <nummer>2</nummer>
    <thema><![CDATA[zusammengesetzte Daten]]&gt;</thema>
    <datei>
      <name>2.sml</name> <punkte>8</punkte>
      <beschreibung>
        <![CDATA[SMalL-Ausdruck <tt>D</tt>]]&gt;
      </beschreibung>
      <vergleich>1</vergleich>
    </datei>
  </aufgabe>

  <aufgabe>
    <nummer>4</nummer>
    <thema><![CDATA[Ausnahmen]]&gt;</thema>
    <datei>
      <name>4.txt</name> <punkte>8</punkte>
      <beschreibung>
        <![CDATA[nixda mit Substitutionsmodell]]&gt;
      </beschreibung>
      <vergleich>1</vergleich>
    </datei>
  </aufgabe>

 </aufgaben>
</uebungsblatt>
```

Wie zu sehen ist, ist für jede der drei Aufgaben ein Element aufgabe vorhanden. Dieses beinhaltet die Aufgabennummer (nummer), das Thema (thema), die dazugehörigen Dateien (in diesem Fall jeweils nur eine) (datei), deren Dateiname (name), die maximal erreichbaren Punkte (punkte), eine kurze Beschreibung (beschreibung) sowie die Angabe der Vergleichs-Stufe (vergleich).

Ist diese Datei erstellt und in das Verzeichnis `conf/xml` kopiert worden, so kann die Generierung der Abgabeseite mit Hilfe des Programms **test_abgabeformular.pl** wie folgt getestet werden:

```
georg@aladdin:~$ /lfe/pms/home/proj/info1korrektur/bin/test_abgabeformular.pl\
  -u 10 > abgabeformular_10.html
```

Ist diese Ausführung erfolgreich, kann die HTML-Datei `abgabeformular_10.html` mit einem Browser geöffnet werden und auf etwaige weitere Fehler hin untersucht werden.

Näheres zur Syntax der Übungsblatt-Konfigurationsdatei findet sich in Abschnitt A.1.1.3.

3.2.1.2. Anlegen der Preload-Datei

Wie oben erwähnt, muss vor dem Syntaxtest bei der Abgabe und der Tests bei der automatischen Verarbeitung der Aufgabe 2 noch eine andere SML-Datei in SML geladen werden.

Dies geschieht durch sogenannte Preload-Dateien.

Diese Dateien werden für die Syntax-Tests im Verzeichnis `preload` im Basis-Verzeichnis der Web-Schnittstelle angelegt. Für Übungsblatt 10 und Datei `2.sml` also wie folgt:

```
info1run@aladdin:/lfe/pms/home/proj/info1run/preload/10$ ls -al
total 10
drwxr-s---    2 info1run info1run     1024 Jan 23 15:40 .
drwxr-s---    3 info1run info1run     1024 Jan 23 15:40 ..
-rw-r-----    1 info1run info1run     7838 Jan 23 15:40 preload-2.sml
```

Für die Tests bei automatischen Verarbeitung der Lösungen muss die Preload-Datei im Testsuite-Verzeichnis liegen:

```
georg@aladdin:/lfe/pms/home/proj/info1korrektur/testsuites/10$ ls -al
total 11
drwxr-s---    2 georg    info1        1024 Jan 10 22:58 .
drwxr-s---   13 georg    info1        1024 Jan 10 22:57 ..
-rwxr-s---    1 georg    info1        7838 Jan 10 22:58 preload-2.sml
```

Die Preload-Datei wird wenn sie exisitiert automatisch bei der Abgabe bzw. beim Test der Lösung geladen.

3.2.1.3. Anlegen der Testsuites

Die Tests werden erst bei der automatischen Verarbeitung der Lösungen durchgeführt, nicht bei der Abgabe. Die Testsuites müssen daher im Gegensatz zu den Preload-Dateien auch nur im Administrations-Verzeichnis angelegt werden.

Um nun Tests mit der Aufgabe 2 durchzuführen ist eine Test-Datei wie folgt anzulegen:

```
georg@aladdin:/lfe/pms/home/proj/info1korrektur/testsuites/10$ ls -al
total 11
drwxr-s---    2 georg    info1        1024 Jan 10 22:58 .
drwxr-s---   13 georg    info1        1024 Jan 10 22:57 ..
-rwxr-s---    1 georg    info1        7838 Jan 10 22:58 preload-2.sml
-rwxr-s---    1 georg    info1         925 Jan 10 22:58 test-2.sml
```

Die Test-Datei wird also im selben Verzeichnis wie die Preload-Datei angelegt und unterscheidet sich von ihr nur im Prefix. Ebenso wie die Preload-Datei automatisch geladen wird, falls sie vorhanden ist, werden die Tests automatisch bei vorhandensein einer Test-Datei für die jeweilige SML-Programmieraufgabe ausgeführt.

3.3. Starten der automatischen Verarbeitung

Ist der Abgabezeitraum für ein Übungsblatt zuende, so kann die weitere Verarbeitung der Aufgaben angestossen werden.

Dies geschieht durch Ausführen des bash-Skripts **startTools.sh** (siehe auch Abschnitt A.3.2.1). Dieses Shell Skript startet verschiedene Perl-Skripte, die die Lösungen der Studenten für die Korrektur vorbereiten. Gestartet wird **startTools.sh** wie folgt:

```
georg@aladdin:/lfe/pms/home/proj/info1korrektur/bin$ ./startTools.sh -u 10
Starte archivierung.pl für 10
Archiv erfolgreich erstellt
Arbeitskopie erfolgreich erstellt
Starte vergleiche.pl für 10
Starte autotest.pl für 10
Starte zusatzinformationen.pl für 10
Starte aufbereiten.pl für 10
Starte zuschicken.pl für 10
Ausführung erfolgreich!
```

Jetzt finden die Korrektoren die von ihnen zu korrigierenden Lösungen in ihrem Verzeichnis und, falls gewünscht, auch im Eingangskorb ihres E-Mail Programms.

3.4. Weitere Adminstrationsaufgaben

Die oben vorgestellten drei Schritte sind die wesentlichen Administrationsaufgaben in direktem Zusammenhang mit dem System. Es existieren jedoch z.B. auch noch Programme wie **latex-db.pl** und **db-latex.pl** zum Im- und Export von Daten aus der Datenbank in eine LaTeX Datei.

Diese Programme sind, wie alle anderen auch, in Anhang A aufgeführt. Mit ihnen können noch einige weitere Adminstrationsaufgaben in Zusammenhang mit dem System bewerkstelligen werden.

Kapitel 4. Installation

In diesem Abschnitt wird auf die Installation des Systems eingegangen. Aspekte dabei sind die Systemvoraussetzungen, die Einrichtung der Benutzer und Benutzergruppen sowie die Anpassungen der Konfiguration.

4.1. Systemvoraussetzung

Die Versionsangaben bei den Softwareprodukten sind jeweils die getesteten Versionen. Neuere Versionen sollten auch funktionieren, ältere können unter Umständen Probleme verursachen.

4.1.1. Perl

Die Version 5.6.0 war zur Zeit der Implementierung die aktuellste Version und war auch auf dem Entwicklungsrechner installiert. Auf dem Produktionsrechner läuft Version 5.005_03, beide Versionen sind also gleichermaßen geeignet.

4.1.1.1. Perl-Module

Alle Module sind, falls nicht bereits in der Perl-Distribution, bei CPAN (www.cpan.org (http://www.cpan.org)) erhältlich. Die Module werden im Allgemeinen mit **`perl Makefile.pl; make; make test; make install`** installiert, näheres findet sich in der Dokumentation zu dem jeweiligen Modul.

- CGI, Version 2.74
- DBI, Version 1.13
- Getopt::Std, Version 1.01
- HTML::Template 2.2
- IO::File, Version 1.06021
- IO::Handle, Version 1.1505
- IPC::Open2, Version 1.01
- MIME::Lite, Version 2.102
- POSIX, Version 1.02
- Time::Local, ohne Versionsangabe
- XML::Parser, Version 2.27

4.1.2. Web-Server

Der Web-Server muss die Ausführung von CGI-Skripten unterstützen. Außerdem wird aus Sicherheitsgründen SSL zur Übertragung der Daten eingesetzt.

4.1.2.1. Apache

Der Apache Server benötigt zur Unterstützung von SSL das Modul mod_ssl (http://www.modssl.org). Des weiteren müssen, ebenso aus Gründen der Sicherheit, die CGI-Skripte unter einer bestimmten Kennung laufen. Dies kann entweder mit Hilfe des Moduls ExecCGI geschehen, oder indem der Apache Server unter eben dieser Benutzer-Kennung läuft.

Im folgenden ein Auschnitt mit den wichtigsten Direktiven aus der Apache Konfiguration httpd.conf. Der Apache Server wird dabei exklusiv für das Abgabesystem benutzt und reagiert nur auf Anfragen per HTTPS.

Beispiel 4-1. Apache Konfiguration - `httpd.conf`

```

##
## httpd.conf -- Apache HTTP server configuration file
##


### Section 1: Global Environment

ServerName info1.informatik.uni-muenchen.de

[...]

LoadModule ssl_module         /usr/lib/apache/libssl.so
LoadModule alias_module       /usr/lib/apache/mod_alias.so

[...]

AddModule mod_alias.c
AddModule mod_ssl.c

[...]

### Section 2: 'Main' server configuration
Port 443
Listen 443

User info1run
Group info1run

DocumentRoot "/lfe/pms/home/proj/info1run/www"
```

```
SetEnv PATH /sbin:/bin:/usr/sbin:/usr/bin:/soft/bin

<Directory />
    Order allow,deny
    Deny from all
    Options -Indexes -FollowSymLinks -Includes -MultiViews
</Directory>

<Directory "/lfe/pms/home/proj/info1run/www">
    Options -Indexes -FollowSymLinks -Includes -MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

[...]

<IfModule mod_alias.c>
    ScriptAlias /cgi-bin/ "/lfe/pms/home/proj/info1run/www/cgi-bin/"
</IfModule>

<Location /cgi-bin>
    AllowOverride None
    Options +ExecCGI -Includes
    SetHandler cgi-script
</Location>

[...]

##
##  SSL Global Context
##

[...]

AddType application/x-x509-ca-cert .crt

[...]

SSLEngine on

[...]

SSLCertificateFile /etc/httpd/ssl.crt/ifilmu.crt

[...]

SSLCertificateKeyFile /etc/httpd/ssl.key/ifilmu.key
```

```
[...]

SSLCertificateChainFile /etc/httpd/ssl.crt/ifilmu.crt

[...]

<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>

<Directory "/lfe/pms/home/proj/info1run/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

[...]
```

Näheres zur Konfiguration findet sich in Dokumentation zum Apache Server unter http://httpd.apache.org/docs/.

4.1.3. DBMS

Zur Verwaltung de Daten wird eine SQL Datenbank benötigt. Die Datenbank muss Locking und Sequences unterstützen. Nicht benötigt werden Sub-Selects oder Transaktionsunterstützung.

4.1.3.1. MySQL

Entwickelt wurde das System unter Verwendung des DBMS MySQL. Näheres zur Einrichtung von MySQL findet sich in Abschnitt 4.2.3 und Abschnitt 4.3. Das MySQL Reference Manual ist unter http://www.mysql.com/documentation/manual.php zu finden.

4.1.4. Sonstige Programme

Es werden die Befehle **rm**, **ln**, **tar**, **cp**, **diff**, **file** und **chmod** benötigt. Außerdem muss eine **bash** zur Verfügung stehen.

4.2. Benutzer und Gruppen

Die verschiedenen Benutzer und Gruppen, die zum Betrieb des Systems eingerichtet werden, dienen durch die Beschränkung der Rechte der Sicherheit. Dabei muss zwischen den Betriebsystem- und Datenbank-Benutzern unterschieden werden.

> **Namen:** Die genannten Namen für die Benutzer sind natürlich frei wählbar. Die in dieser Installationsanweisung verwendeten Benutzernamen entsprechen den Namen der Benutzer für die Installation des Systems im Wintersemester 2000/2001 für die Vorlesung "Informatik I".

4.2.1. Betriebssystem

Auf Betriebssystem-Ebene sind, um den Sicherheitsanforderungen an das Systems gerecht zu werden, drei verschiedene Benutzer und bzw. Gruppen notwendig.

4.2.1.1. Benutzer und Gruppe info1run

Unter diesem Benutzer und unter dieser Gruppe läuft der Web-Server. Alle direkt durch die Web-Schnittstelle angestossenen Programme und auch alle Dateien, die direkt zur Web-Schnittstelle gehören, haben den Besitzer und die Gruppe info1run.

Der Web-Server kann auch ohne Probleme unter dem jeweiligen Standard-Benutzer für den Web-Server (www-data bei Debian, wwwrun bei SuSE) laufen. Die Dateien der Web-Schnittstelle erhalten dann diesen Benutzer, die Gruppe muss jedoch weiterhin info1run sein.

4.2.1.1.1. Mitglieder der Gruppe

info1run. info1run muss nicht Mitglied der Gruppe info1run sein.

Administrator. Der Administrator muss Mitglied dieser Gruppe sein, um Zugriff auf die abgegeben Übungsaufgaben und auf die XML-Konfigurationsdateien für die Übungsblätter zu haben. Auf diese Dateien wird durch das Skript **startTools.sh** zugegriffen.

4.2.1.2. Benutzer und Gruppe info1abg

Dieser Benutzer und diese Gruppe dient einzig für die Ausführung von durch die Studenten abgegebenen Programmen. Dies erhöht die Sicherheit vor Auswirkungen von beabsichtigten und unbeabsichtigten Eigenschaften der Programme auf das System.

4.2.1.2.1. Mitglieder der Gruppe

info1abg. In dieser Gruppe ist kein Benutzer außer info1abg Mitglied.

4.2.1.3. Gruppe info1

Unter dieser Gruppenkennung verläuft die automatische Verarbeitung der abgegebenen Dateien. Die zur Korrektur aufbereiteten Übungsaufgaben gehören auch dieser Benutzergruppe. Daher müssen alle Korrektoren als auch der Administrator, der den Prozess zur automatischen Aufbereitung anstößt, Mitglieder dieser Gruppe sein.

4.2.1.3.1. Mitglieder der Gruppe

Korrektoren. Die Korrektoren sind Mitglieder dieser Gruppe, um Zugriff auf die zu korrigierenden Dateien zu erlangen und die korrigierten Dateien mit dem Skript korrigiert.sh abgeben zu können.

Administrator. Der Administrator ist Mitglied dieser Gruppe, um die automatische Verarbeitung der abgegebenen Übungsaufgaben starten zu können.

4.2.2. E-Mail Adressen

Die Studenten und die Korrektoren erhalten vom System automatisch E-Mails. Für diese E-Mails muss ein Absender existieren. Sinnvoll ist es auch, dass die Studenten bei Problemen mit dem System direkt an diese Adresse schreiben können.

Für die Vorlesung "Informatik I" im WS 2000/2001 wurde zu diesem Zweck die E-Mail Adresse `<info1@informatik.uni-muenchen.de>` eingerichtet. Diese E-Mail Adresse muss auch in Konfiguration.pm - Web (Abschnitt A.1.1.1) und Tool (Abschnitt A.2.1.1) - eingetragen werden.

4.2.3. Datenbank

Auch für die Datenbank sind aus Sicherheits-Aspekten verschiedene Benutzer mit unterschiedlichen Rechten nötig. Die Benutzer werden bei MySQL selbst in der Datenbank mysql, Tabelle user, verwaltet. Bei Verbindung mit der Datenbank mysql legen die folgenden SQL-Anweisungen die einzelnen Benutzer an. Das Passwort sollte natürlich verändert werden.

4.2.3.1. Benutzer info1admin

Der Benutzer info1admin ist der Administrator der Datenbank. Er hat alle Rechte für die Datenbank.

```
INSERT INTO user (Host, User, Password) \
  VALUES ('localhost','info1admin',PASSWORD('Sonne'));
```

4.2.3.2. Benutzer info1tool

info1tool dient für den Datenbankzugriff durch die Werkzeuge.

```
INSERT INTO user (Host, User, Password) \
  VALUES ('localhost','info1tool',PASSWORD('Mond'));
```

4.2.3.3. Benutzer info1run

Der Datenbankzugriff durch Web-Schnittstelle erfolgt mit dem Benutzer info1run.

```
INSERT INTO user (Host, User, Password) \
  VALUES ('localhost','info1run',PASSWORD('Sterne'));
```

4.3. Einrichten der Datenbank

Nach Installation des Datenbanksystems muss dieses noch für die Benutzung durch das System eingerichtet werden. Dazu sind zum einen die verschiedenen Benutzer mit ihren Zugriffsrechten einzurichten, als auch die Datenbank für das System angelegt werden.

4.3.1. Zugriffsrechte

Den verschiedenen Benutzern der Datenbank müssen für die verschiedenen Tabellen unterschiedliche Rechte wie SELECT, UPDATE und INSERT zugewiesen werden. Bei MySQL geschieht dies durch die beiden Tabellen db und user_priv in der Datenbank mysql.

In der Tabelle db werden dem Benutzer info1admin alle Rechte für die Datenbank info1 gegeben. Den anderen Benutzern werden durch diese Tabelle alle Rechte verweigert. Ihre Rechte erhalten sie aufgrund der Einträge in der Tabelle user_priv. Dort ist eine Abstufung der Rechte auf Tabellenebene möglich.

```
INSERT INTO db (Host, Db, User) \
  VALUES ('localhost','info1','info1tool');

INSERT INTO db (Host, Db, User) \
  VALUES ('localhost','info1','info1run');

INSERT INTO db (Host, Db, User, Select_priv, Insert_priv, \
  Update_priv, Delete_priv, Create_priv, Drop_priv, \
  Grant_priv, References_priv, Index_priv, Alter_priv) \
```

```
  VALUES ('localhost','info1','info1admin','Y','Y','Y', \
  'Y','Y','Y','Y','Y','Y','Y');


INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1run','abgabe','Select,Insert');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1run','aufgabe','Select');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1run','korrektor','Select');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1run','punkte','Select,Insert');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1run','student','Select,Insert,Update');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1run','tausch','Select,Delete');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1run','uebungsgruppe','Select');

INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1tool','abgabe','Select,Insert');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1tool','aufgabe','Select,Insert');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1tool','klausur','Select');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1tool','klausur_student','Select,Insert,Update');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1tool','korrektor','Select');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1tool','punkte','Select,Insert,Update');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1tool','student','Select');
INSERT INTO tables_priv (Host, Db, User, Table_name, Table_priv) \
  VALUES ('localhost','info1','info1tool','uebungsgruppe','Select');
```

4.3.2. Initialisierung

Zuerst muss die Datenbank für die Veranstaltung angelegt werden, z.B. info1 für die Vorlesung "Informatik I":

```
CREATE DATABASE info1;
```

Für die folgenden Befehle muss man diese neu angelegte Datenbank benutzen. Bei MySQL geschieht dies mit:

```
\u info1
```

Im letzten Schritt müssen jetzt noch die Tabellen angelegt werden. Dies geschieht mit den folgenden SQL-Anweisungen.

```
#
# Table structure for table 'abgabe'
#
CREATE TABLE abgabe (
  abgabe_id int(11) DEFAULT '0' NOT NULL auto_increment,
  abgabe_teamnr int(11) DEFAULT '0' NOT NULL,
  uebungsblatt_nummer tinyint(4) DEFAULT '0' NOT NULL,
  student_kennung varchar(8) DEFAULT '' NOT NULL,
  PRIMARY KEY (abgabe_id)
);

#
# Table structure for table 'aufgabe'
#
CREATE TABLE aufgabe (
  aufgabe_id int(11) DEFAULT '0' NOT NULL auto_increment,
  aufgabe_dateiname varchar(50) DEFAULT '' NOT NULL,
  uebungsblatt_nummer int(11) DEFAULT '0' NOT NULL,
  PRIMARY KEY (aufgabe_id)
);

#
# Table structure for table 'aufgabe_teamnr'
#
CREATE TABLE aufgabe_teamnr (
  aufgabe_teamnr_id int(11) DEFAULT '0' NOT NULL auto_increment,
  aufgabe_id int(11) DEFAULT '0' NOT NULL,
  abgabe_teamnr int(11) DEFAULT '0' NOT NULL,
  PRIMARY KEY (aufgabe_teamnr_id)
);

#
# Table structure for table 'klausur'
#
CREATE TABLE klausur (
  klausur_nummer tinyint(4) DEFAULT '0' NOT NULL,
  PRIMARY KEY (klausur_nummer)
);

#
# Table structure for table 'klausur_student'
#
CREATE TABLE klausur_student (
  klausur_student_id int(11) DEFAULT '0' NOT NULL auto_increment,
```

```
  klausur_student_zulassung enum('nicht zugelassen','fraglich','zugelassen'),
  klausur_student_punkte tinyint(4),
  klausur_student_note float(10,2),
  klausur_student_ergebnis enum('nicht teilgenommen','nicht bestanden',\
    'fraglich','bestanden'),
  klausur_nummer tinyint(4) DEFAULT '0' NOT NULL,
  student_kennung varchar(8) DEFAULT '' NOT NULL,
  PRIMARY KEY (klausur_student_id)
);

#
# Table structure for table 'korrektor'
#
CREATE TABLE korrektor (
  korrektor_name varchar(50),
  korrektor_vorname varchar(50),
  korrektor_email varchar(50),
  korrektor_aufgabenzuschicken tinyint(1) DEFAULT '0',
  korrektor_kennung varchar(8) DEFAULT '' NOT NULL,
  KEY korrektor_kennung (korrektor_kennung)
);

#
# Table structure for table 'punkte'
#
CREATE TABLE punkte (
  punkte_id int(11) DEFAULT '0' NOT NULL auto_increment,
  punkte int(11) DEFAULT '0' NOT NULL,
  aufgabe_id int(11) DEFAULT '0' NOT NULL,
  student_kennung varchar(8) DEFAULT '' NOT NULL,
  PRIMARY KEY (punkte_id)
);

#
# Table structure for table 'student'
#
CREATE TABLE student (
  student_name varchar(50),
  student_vorname varchar(50),
  student_matrikelnummer varchar(12),
  student_hauptfach varchar(50),
  student_kennung varchar(8) DEFAULT '' NOT NULL,
  uebungsgruppe_id int(11),
  korrektor_kennung varchar(8),
  PRIMARY KEY (student_kennung)
);

#
# Table structure for table 'tausch'
```

```
#
CREATE TABLE tausch (
  tausch_kennwort varchar(10)
);

#
# Table structure for table 'uebungsblatt'
#
CREATE TABLE uebungsblatt (
  uebungsblatt_nummer tinyint(4) DEFAULT '0' NOT NULL,
  PRIMARY KEY (uebungsblatt_nummer)
);

#
# Table structure for table 'uebungsgruppe'
#
CREATE TABLE uebungsgruppe (
  uebungsgruppe_id int(11) DEFAULT '0' NOT NULL auto_increment,
  uebungsgruppe_maxstudenten tinyint(4) DEFAULT '0' NOT NULL,
  leiter_kennung varchar(8),
  default_korrektor_kennung varchar(8),
  uebungsgruppe_spitzname varchar(15),
  uebungsgruppe_zeit varchar(15),
  uebungsgruppe_ort varchar(15),
  PRIMARY KEY (uebungsgruppe_id)
);
```

Vor der Benutzung des Systems muss die Datenbank noch mit einigen Informationen, z.B. über die Übungsgruppen, gefüllt werden. Auf die dazu notwendigen Schritte wird in Kapitel 3 eingegangen.

4.4. Web-Schnittstelle

Im folgenden wird die Installation der Web-Schnittstelle exemplarisch für das Wintersemester 2000/2001 beschrieben. Falls eine andere Programmiersprache als SML zum Einsatz kommt, müssen mehrere Änderungen an den Programmen, nicht nur bei der Web-Schnittstelle, vorgenommen werden. Auf diese Änderungen wird in dieser Installationsanweisung nicht näher eingegangen.

4.4.1. Verzeichnisstruktur und Rechte von `/home/proj/info1run`

Das Verzeichnis `/home/proj/info1run` ist das Basis Verzeichnis der Web-Schnittstelle. Unter diesem Verzeichnis befinden sich folgende vier Unterverzeichnisse, selbst teilweise wieder mit Unterverzeichnissen:

- `abgabe`: Speicherung der Abgegebene Aufgaben
- `bin`: Nicht CGI-Programme der Web-Schnittstelle
- `conf`: Konfiguration der Web-Schnittstelle
  - `tpl`: HTML-Vorlagen
  - `xml`: XML-Konfigurationsdateien für die Übungsblätter
- `korrektur`: Speicherung der korrigierten Aufgaben
  - `arbeit`: Arbeits-Verzeichnis für die Entgegennahme der korrigierten Aufgaben über die Web-Schnittstelle
  - `archiv`: Archiv-Verzeichnis für die Speicherung und Archivierung der über die Web-Schnittstelle zurückgegebene korrigierten Aufgaben
- `preload`: SML-Programme, die vor den von den Studenten abgegebenen Programme in SML geladen werden müssen, um die Programme ausführen zu können. Unterhalb dieses Verzeichnisses werden im Laufe des Semesters für die einzelnen Übungsblätter Unterverzeichnisse angelegt.
- `www`: statische HTML-Seiten
  - `cgi-bin`: CGI-Skripte zur dynamischen Generierung von HTML-Seiten und zur Verarbeitung von Formulardaten

Die folgende Verzeichnis-Ansicht gibt die genaue Struktur und die Zugriffsrechte der Verzeichnisse wieder.

Im Wintersemester 2000/2001 ist `/home/proj/info1run` ein symbolischer Link auf `/lfe/pms/home/proj/info1run`.

```
info1run@aladdin:~$ pwd
/lfe/pms/home/proj/info1run
info1run@aladdin:~$ ls -alR
.:
total 12
drwxr-x---    7 info1run info1run     1024 Jan  7 18:28 .
drwxr-xr-x    4 root     root         1024 Jan  7 18:17 ..
[...]
drwxr-x---    2 info1run info1run     1024 Jan  7 18:27 abgabe
drwx------    2 info1run info1run     1024 Jan  7 18:27 bin
drwxr-x---    4 info1run info1run     1024 Jan  7 18:29 conf
drwx------    4 info1run info1run     1024 Jan  7 18:30 korrektur
```

```
drwx------    4 info1run info1run     1024 Jan  7 18:30 preload
drwx------    3 info1run info1run     1024 Jan  7 18:29 www

./abgabe:
total 2
drwxr-x---    2 info1run info1run     1024 Jan  7 18:27 .
drwxr-x---    7 info1run info1run     1024 Jan  7 18:28 ..

./bin:
total 2
drwx------    2 info1run info1run     1024 Jan  7 18:27 .
drwxr-x---    7 info1run info1run     1024 Jan  7 18:28 ..

./conf:
total 4
drwxr-x---    4 info1run info1run     1024 Jan  7 18:29 .
drwxr-x---    7 info1run info1run     1024 Jan  7 18:28 ..
drwxr-x---    2 info1run info1run     1024 Jan  7 18:29 tpl
drwxr-x---    2 info1run info1run     1024 Jan  7 18:29 xml

./conf/tpl:
total 2
drwxr-x---    2 info1run info1run     1024 Jan  7 18:29 .
drwxr-x---    4 info1run info1run     1024 Jan  7 18:29 ..

./conf/xml:
total 2
drwxr-x---    2 info1run info1run     1024 Jan  7 18:29 .
drwxr-x---    4 info1run info1run     1024 Jan  7 18:29 ..

./korrektur:
total 4
drwx------    4 info1run info1run     1024 Jan  7 18:30 .
drwxr-x---    7 info1run info1run     1024 Jan  7 18:28 ..
drwx------    2 info1run info1run     1024 Jan  7 18:29 arbeit
drwx------    2 info1run info1run     1024 Jan  7 18:30 archiv

./korrektur/arbeit:
total 2
drwx------    2 info1run info1run     1024 Jan  7 18:29 .
drwx------    4 info1run info1run     1024 Jan  7 18:30 ..

./korrektur/archiv:
total 2
drwx------    2 info1run info1run     1024 Jan  7 18:30 .
drwx------    4 info1run info1run     1024 Jan  7 18:30 ..

./www:
total 3
```

```
drwx------    3 info1run info1run     1024 Jan  7 18:29 .
drwxr-x---    7 info1run info1run     1024 Jan  7 18:28 ..
drwx------    2 info1run info1run     1024 Jan  7 18:29 cgi-bin

./www/cgi-bin:
total 2
drwx------    2 info1run info1run     1024 Jan  7 18:29 .
drwx------    3 info1run info1run     1024 Jan  7 18:29 ..
```

4.4.2. Programme und Dateien

4.4.2.1. `bin`

In diesem Verzeichnis befinden sich die beiden Programme **bewertungen.pl** (Abschnitt A.1.4.2) und **punktestand.pl** (Abschnitt A.1.4.3).

```
info1run@aladdin:~/bin$ pwd
/lfe/pms/home/proj/info1run/bin
info1run@aladdin:~/bin$ ls -al
total 21
drwx------    2 info1run info1run     1024 Jan  7 18:49 .
drwxr-x---    7 info1run info1run     1024 Jan  7 18:28 ..
-rwx------    1 info1run info1run    11250 Jan  7 18:49 bewertungen.pl
-rwx------    1 info1run info1run     8139 Jan  7 18:49 punktestand.pl
```

4.4.2.2. `conf/tpl`

Die Dateien in diesem Verzeichnis dienen als Vorlagen für die dynamische Generierung der HTML-Seiten durch die CGI-Skripte. Diese Dateien müssen für andere Veranstaltungen jeweils angepasst werden. Dabei müssen die Kommentare der Form `<!-- TMPL_* * -->` erhalten bleiben. Diese Kommentare werden durch die CGI-Programme dynamisch ersetzt, mit diesem Mechanismus werden z.B. die beiden Vorlagen `kopf.html` und `fuss.html` von den anderen Vorlagen als Kopf und Fuß des resultierenden HTML-Dokuments eingebunden.

```
info1run@aladdin:~/conf/tpl$ pwd
/lfe/pms/home/proj/info1run/conf/tpl
info1run@aladdin:~/conf/tpl$ ls -al
total 33
drwxr-x---    2 info1run info1run     1024 Jan  7 18:52 .
drwxr-x---    4 info1run info1run     1024 Jan  7 18:29 ..
-rw-r-----    1 info1run info1run     1571 Jan  7 18:52 abgabefehler.html
-rw-r-----    1 info1run info1run     4488 Jan  7 18:52 abgabeformular.html
-rw-r-----    1 info1run info1run     1507 Jan  7 18:52 abgabeok.html
-rw-r-----    1 info1run info1run     4456 Jan  7 18:52 anmeldeformular.html
```

```
-rw-r-----    1 info1run info1run     1415 Jan  7 18:52 anmeldung.html
-rw-r-----    1 info1run info1run     2458 Jan  7 18:52 belegung.html
-rw-r-----    1 info1run info1run     1843 Jan  7 18:52 fehlendekonfiguration.html
-rw-r-----    1 info1run info1run       19 Jan  7 18:52 fuss.html
-rw-r-----    1 info1run info1run     1488 Jan  7 18:52 keineabgabe.html
-rw-r-----    1 info1run info1run     1221 Jan  7 18:52 kopf.html
-rw-r-----    1 info1run info1run     1422 Jan  7 18:52 korrekturfehler.html
-rw-r-----    1 info1run info1run     1752 Jan  7 18:52 korrekturok.html
-rw-r-----    1 info1run info1run     1470 Jan  7 18:52 matrikelnummer.html
-rw-r-----    1 info1run info1run     1466 Jan  7 18:52 tausch.html
```

4.4.2.3. `conf/xml`

Die Dateien in diesem Verzeichnis (bis auf uebungsblaetter.xml, siehe Abschnitt A.1.1.2) werden erst im Laufe des Semesters angelegt. Die folgende Ausgabe von **ls** zeigt das Aussehen mitten im Semester.

```
info1run@aladdin:~/conf/xml$ pwd
/lfe/pms/home/proj/info1run/conf/xml
info1run@aladdin:~/conf/xml$ ls -al
total 28
drwxr-x---    2 info1run info1run     1024 Jan  7 18:59 .
drwxr-x---    4 info1run info1run     1024 Jan  7 18:29 ..
-rw-r-----    1 info1run info1run     2109 Jan  7 18:59 uebungsblaetter.xml
-rw-r-----    1 info1run info1run     2868 Jan  7 18:59 uebungsblatt01.xml
-rw-r-----    1 info1run info1run     1793 Jan  7 18:59 uebungsblatt02.xml
-rw-r-----    1 info1run info1run     2305 Jan  7 18:59 uebungsblatt03.xml
-rw-r-----    1 info1run info1run     2517 Jan  7 18:59 uebungsblatt04.xml
-rw-r-----    1 info1run info1run     2037 Jan  7 18:59 uebungsblatt05.xml
-rw-r-----    1 info1run info1run     2321 Jan  7 18:59 uebungsblatt06.xml
-rw-r-----    1 info1run info1run     1519 Jan  7 18:59 uebungsblatt07.xml
-rw-r-----    1 info1run info1run     1479 Jan  7 18:59 uebungsblatt08.xml
-rw-r-----    1 info1run info1run     2305 Jan  7 18:59 uebungsblatt09.xml
```

Auf die Erstellung dieser Dateien wird in Kapitel 3 eingegangen.

4.4.2.4. `www`

Die HTML-Dateien in diesem Verzeichnis sind Formulare. Sie müssen, ebenso wie bei den Templates, für andere Veranstaltungen angepasst werden.

```
info1run@aladdin:~/www$ pwd
/lfe/pms/home/proj/info1run/www
info1run@aladdin:~/www$ ls -al
total 13
drwx------    3 info1run info1run     1024 Jan  7 19:01 .
```

```
drwxr-x---    7 info1run info1run     1024 Jan  7 18:28 ..
drwx------    2 info1run info1run     1024 Jan  7 18:29 cgi-bin
-rw-------    1 info1run info1run     2534 Jan  7 19:01 korrekturformular.html
-rw-------    1 info1run info1run     2676 Jan  7 19:01 matrikelnummerformular.html
-rw-------    1 info1run info1run     3601 Jan  7 19:01 tauschformular.html
```

4.4.2.5. `www/cgi-bin`

In diesem Verzeichnis liegen die Perl-Skripte für die Web-Schnittstelle. Angepasst werden muss hier nur die Datei `Konfiguration.pm` (Abschnitt A.1.1.1), welche die zentrale Konfigurationsdatei für die Perl-Skripte der Web-Schnittstelle darstellt.

```
info1run@aladdin:~/www/cgi-bin$ pwd
/lfe/pms/home/proj/info1run/www/cgi-bin
info1run@aladdin:~/www/cgi-bin$ ls -al
total 93
drwx------    2 info1run info1run     1024 Jan  7 19:12 .
drwx------    3 info1run info1run     1024 Jan  7 19:01 ..
-rw-------    1 info1run info1run     2574 Jan  7 19:12 Konfiguration.pm
-rwx------    1 info1run info1run    20016 Jan  7 19:12 abgabe.pl
-rwx------    1 info1run info1run    10295 Jan  7 19:12 abgabeformular.pl
-rwx------    1 info1run info1run     6474 Jan  7 19:12 anmeldeformular.pl
-rwx------    1 info1run info1run    14950 Jan  7 19:12 anmeldung.pl
-rwx------    1 info1run info1run     6152 Jan  7 19:12 belegung.pl
-rwx------    1 info1run info1run     8532 Jan  7 19:12 korrektur.pl
-rwx------    1 info1run info1run     7332 Jan  7 19:12 matrikelnummer.pl
-rwx------    1 info1run info1run     9100 Jan  7 19:12 tausch.pl
```

4.5. Werkzeuge

Ebenso wie bei der Installation der Web-Schnittstelle wird auch an dieser Stelle die die Installation exemplarisch für das Wintersemester 2000/2001 - Vorlesung "Informatik I" - durchgeführt.

4.5.1. Verzeichnisstruktur und Rechte von `/home/proj/info1korrektur`

Das Verzeichnis `/home/proj/info1korrektur` ist das Basis Verzeichnis der Administrations-Werkzeuge sowie für die Korrektur der Übungsaufgaben. Unter diesem Verzeichnis befinden sich folgende fünf Unterverzeichnisse, selbst teilweise wieder mit Unterverzeichnissen:

- `archiv`: Archvierung der Übungsaufgaben (noch nicht korrigiert, sondern genau wie abgegeben)
- `bin`: Programme zur Administration und zur automatischen Verarbeitung der Übungsaufgaben
- `korrektor`: Korrektor-Verzeichnis. In diesem Verzeichnis wird automatisch für jeden Korrektor ein Unterverzeichnis angelegt, in dem der Korrektor jeweils die zu korrigierenden Aufgaben findet.
- `testsuites`: Verzeichnis mit Beschreibungen der automatisch durchzuführenden Tests und eventuell vor dem abgegebenen Programm zu ladenden Programmcode.
- `tmp`: Verzeichnis für temporäre Dateien
  - `arbeit`: Verzeichnis für temporär während der automatischen Verarbeitung der Übungsaufgaben angelegten Dateien

Die Rechte in diesem Verzeichnis können gegenüber dem unten folgenden Beispiel auch lockerer gehandhabt werden. In dem folgenden Beispiel hat nur der Administrator (hier: georg) die Schreibrechte, sind jedoch mehrere Personen mit der Administration beauftragt, kann es sinnvoll sein, der gesamten Gruppe info1 Schreibrechte für bestimmte Verzeichnisse zu gewähren. In diesem Fall ist es jedoch zu überlegen, ob nicht ein gesondert eingerichteter Benutzer als Administrator sinnvoller ist.

```
georg@aladdin:/lfe/pms/home/proj/info1korrektur$ pwd
/lfe/pms/home/proj/info1korrektur
georg@aladdin:/lfe/pms/home/proj/info1korrektur$ ls -alR
.:
total 7
drwxr-s---    7 georg    info1        1024 Jan 15 23:48 .
drwxr-xr-x    5 root     root         1024 Jan 10 22:57 ..
drwxr-s---    2 georg    info1        1024 Jan 15 23:41 archiv
drwxr-s---    2 georg    info1        1024 Jan 11 21:32 bin
drwxr-s---    2 georg    info1        1024 Jan 15 23:42 korrektor
drwxr-s---   13 georg    info1        1024 Jan 10 22:57 testsuites
drwxr-s---    3 georg    info1        1024 Jan 15 23:48 tmp

./archiv:
total 2
drwxr-s---    2 georg    info1        1024 Jan 15 23:41 .
drwxr-s---    7 georg    info1        1024 Jan 15 23:48 ..

./bin:
total 107
drwxr-s---    2 georg    info1        1024 Jan 11 21:32 .
drwxr-s---    7 georg    info1        1024 Jan 15 23:48 ..
```

```
./korrektor:
total 2
drwxr-s---    2 georg    info1        1024 Jan 15 23:42 .
drwxr-s---    7 georg    info1        1024 Jan 15 23:48 ..

./testsuites:
total 13
drwxr-s---   13 georg    info1        1024 Jan 10 22:57 .
drwxr-s---    7 georg    info1        1024 Jan 15 23:48 ..

./tmp:
total 3
drwxr-s---    3 georg    info1        1024 Jan 15 23:48 .
drwxr-s---    7 georg    info1        1024 Jan 15 23:48 ..
drwxr-s---    2 georg    info1        1024 Jan 15 23:48 arbeit

./tmp/arbeit:
total 2
drwxr-s---    2 georg    info1        1024 Jan 15 23:48 .
drwxr-s---    3 georg    info1        1024 Jan 15 23:48 ..
```

4.5.2. Programme und Dateien

4.5.2.1. `bin`

In diesem Verzeichnis befinden sich die Werkzeuge zur Administration.

Angepasst werden muss in diesem Verzeichnis die Datei `Konfiguration.pm` (Abschnitt A.2.1.1).

```
georg@aladdin:/lfe/pms/home/proj/info1korrektur/bin$ pwd
/lfe/pms/home/proj/info1korrektur/bin
georg@aladdin:/lfe/pms/home/proj/info1korrektur/bin$ ls -al
total 107
drwxr-s---    2 georg    info1        1024 Jan 11 21:32 .
drwxr-s---    7 georg    info1        1024 Jan 15 23:48 ..
-rwxr-s---    1 georg    info1        3005 Jan 10 22:57 Konfiguration.pm
-rwxr-s---    1 georg    info1        3567 Jan 10 22:57 archivierung.pl
-rwxr-s---    1 georg    info1        7882 Jan 10 22:57 aufbereiten.pl
-rwxr-s---    1 georg    info1        3446 Jan 10 22:57 autotest.pl
-rwxr-s---    1 georg    info1       11217 Jan 10 22:57 bewertungen.pl
-rwxr-s---    1 georg    info1        9500 Jan 10 22:57 db-latex.pl
-rwxr-s---    1 georg    info1        4965 Jan 10 22:57 korrekturstand.pl
-rwxr-s---    1 georg    info1        6691 Jan 10 22:57 latex-db.pl
-rwxr-s---    1 georg    info1        1611 Jan 10 22:57 mail-matrikelnummernachtrag.pl
-rwxr-s---    1 georg    info1       10928 Jan 10 22:57 nachkorrektur.pl
-rwxr-s---    1 georg    info1        8151 Jan 10 22:57 punktestand.pl
-rwxr-s---    1 georg    info1        1970 Jan 10 22:57 startTools.sh
```

```
-rwxr-s---    1 georg    info1        8068 Jan 10 22:57 test_abgabeformular.pl
-rwxr-s---    1 georg    info1        4164 Jan 11 00:46 testen.pl
-rwxr-s---    1 georg    info1        5612 Jan 10 22:57 vergleiche.pl
-rwxr-s---    1 georg    info1        5943 Jan 10 22:57 zusatzinformationen.pl
-rwxr-s---    1 georg    info1        4119 Jan 10 22:57 zuschicken.pl
```

4.6. Konfiguration des Aufrufs von SML

Die Programme der Studenten können Schaden verursachen. Aus diesem Grund laufen sie unter einer besonderen Benutzerkennung und mit weiteren Sicherheitsmaßnahmen wie einer Zeit- und Speicherbeschränkung.

Die folgende Konfiguration ist zwar spezifisch für SML, eine ähnliche Konfiguration muss jedoch auch für andere Programmiersprachen verwendet werden.

4.6.1. Verzeichnisstruktur, Rechte und Dateien unter `/home/proj/info1abg`

In diesem Verzeichnis müssen die Pfade in den beiden bash-Skripten `ulimit_restricted_SML.sh` und `ulimit_restricted_bash_SML.sh` angepasst werden.

```
info1abg@aladdin:~$ pwd
/lfe/pms/home/proj/info1abg
info1abg@aladdin:~$ ls -alR
.:
total 11
drwxr-xr-x    4 info1abg info1abg     1024 Jan 16 00:14 .
drwxr-xr-x    5 root     root         1024 Jan 10 22:57 ..
drwsr-sr-x    2 info1abg info1abg     1024 Jan 16 00:14 bin

./bin:
total 18
drwsr-sr-x    2 info1abg info1abg     1024 Jan 16 00:14 .
drwxr-xr-x    4 info1abg info1abg     1024 Jan 16 00:14 ..
-rwsr-sr-x    1 info1abg info1abg    13035 Jan 10 18:17 timelimit
-rwsr-sr-x    1 info1abg info1abg       71 Jan 16 00:14 ulimit_restricted_SML.sh
-rwsr-sr-x    1 info1abg info1abg      103 Jan 16 00:14 ulimit_restricted_bash_SML.sh
```

Anhang A. Programme und wichtige Dateien

In diesem Abschnitt werden kurz alle Programme vorgestellt. Auch wird auf die wichtigsten Konfigurationsdateien eingegangen.

A.1. Web-Schnittstelle

Über die Web-Schnittstelle können die Studenten sich zu den Übungsgruppen anmelden und die Übungsaufgaben abgeben. Die Korrektoren können die Web-Schnittstelle benutzen, um ihre korrigierten Aufgaben wieder an das System zurückzugeben.

A.1.1. Konfigurationsdateien

Bei der Web-Schnittstelle kommen zwei verschiedene Typen von Konfigurationsdateien zum Einsatz. Zum einen ist das die Datei `www/cgi-bin/Konfiguration.pm`, ein Perl-Modul, welches einmalig bei der Installation des Systems angepasst werden muss. Zum anderen sind es die Dateien `conf/xml/uebungsblaetter.xml` und `conf/xml/uebungsblatt??.xml`. Diese Dateien werden während des Semesters für die einzelnen Übungsblätter erstellt.

A.1.1.1. `www/cgi-bin/Konfiguration.pm`

Zentrale Konfigurationsdatei für die Perl Programme der Web-Schnittstelle.

A.1.1.1.1. Aufgabe

Konfiguration aller von der jeweiligen Installation abhängigen von den Perl Programmen benötigter Werte wie z.B. Verzeichnis- und Dateinamen.

A.1.1.1.2. Beispiel und Erläuterung

Beispiel A-1. Datei `www/cgi-bin/Konfiguration.pm`

```
#!/usr/bin/perl -w

############################################################################
#Name:  Konfiguration.pm
#Projekt: Projektarbeit - Einsatz elektronischer Medien
#  für den Übungsbetrieb
#Sinn&amp;Zweck: Konfiguration der Web-Schnittstelle
#Autor:  Georg Schneemayer
############################################################################
```

```

use strict;

package Konfiguration;

# Pfade                                                    ❶
$Konfiguration::pfad_webbasis = "/home/proj/info1run/";
$Konfiguration::pfad_konfiguration = "conf/xml/";
$Konfiguration::pfad_vorlage = "conf/tpl/";
$Konfiguration::pfad_korrektur = "korrektur/";
$Konfiguration::pfad_korrektur_arbeit = "arbeit/";
$Konfiguration::pfad_korrektur_archiv = "archiv/";
$Konfiguration::pfad_abgabe = "abgabe/";
$Konfiguration::pfad_bin = "bin/";
$Konfiguration::pfad_cgibin = "www/cgi-bin/";
$Konfiguration::pfad_preload = "preload/";

#Dateien
#-Vorlagen                                                 ❷
$Konfiguration::datei_vorlage_anmeldeformular = "anmeldeformular.html";
$Konfiguration::datei_vorlage_anmeldung = "anmeldung.html";
$Konfiguration::datei_vorlage_abgabeformular = "abgabeformular.html";
$Konfiguration::datei_vorlage_keineabgabe = "keineabgabe.html";
$Konfiguration::datei_vorlage_abgabefehler = "abgabefehler.html";
$Konfiguration::datei_vorlage_abgabeok = "abgabeok.html";
$Konfiguration::datei_vorlage_belegung = "belegung.html";
$Konfiguration::datei_vorlage_matrikelnummer = "matrikelnummer.html";
$Konfiguration::datei_vorlage_fehlendekonfiguration = "fehlendekonfiguration.html";
$Konfiguration::datei_vorlage_korrekturfehler = "korrekturfehler.html";
$Konfiguration::datei_vorlage_korrekturok = "korrekturok.html";
$Konfiguration::datei_vorlage_tausch = "tausch.html";
#-Konfiguration                                            ❸
$Konfiguration::datei_konfiguration_uebungsblaetter = "uebungsblaetter.xml";
$Konfiguration::datei_konfiguration_uebungsblatt_muster = "uebungsblattYXC.xml";
$Konfiguration::datei_konfiguration_uebungsblatt_muster_ersetzen = "YXC";

#Konfiguration                                             ❹
$Konfiguration::konfiguration_db_benutzer = "info1run";
$Konfiguration::konfiguration_db_passwort = "YXCVBNM";
$Konfiguration::konfiguration_db_name = "info1";
$Konfiguration::konfiguration_belegungsfaktor = "85";
$Konfiguration::konfiguration_sml_bin = \
                "/lfe/pms/home/proj/info1abg/bin/restricted_bash_SML.sh";
$Konfiguration::konfiguration_perl_bin = "/usr/bin/perl";
$Konfiguration::konfiguration_prefix_preload = "preload-";
$Konfiguration::konfiguration_url_home = \
                "http://www.pms.informatik.uni-muenchen.de/lehre/info1/00ws01/";
$Konfiguration::konfiguration_admin_email = "info1\@informatik.uni-muenchen.de";
$Konfiguration::konfiguration_veranstaltung = "Informatik I (WS 2000/2001)";
```

```
$Konfiguration::konfiguration_gruss = "Grüße,\n\n Info I-Team";
```

❶ An dieser Stelle müssen die Pfade für die Web-Schnittstelle eingetragen werden. Im einzelnen sind dies:

`webbasis`
: Basis-Verzeichnis für die Web-Schnittstelle. Alle anderen Pfade sind relativ zu diesem Verzeichnis.

`konfiguration`
: Unter diesem Verzeichnis finden sich die XML-Konfigurations-Dateien für die Übungsblätter.

`vorlage`
: Verzeichnis mit den Vorlagen für die dynamisch generierten Seiten.

`korrektur`
: Verzeichnis, in dem die über die Web-Schnittstelle zurückgegeben korrigierten Aufgaben verarbeitet und archiviert werden.

  `arbeit`
  : Verzeichnis für temporäre Dateien bei der Verarbeitung der korrigierten Aufgaben.

  `archiv`
  : Verzeichnis für die Archivierung der über die Web-Schnittstelle zurückgegebenen korrigierten Aufgaben.

`abgabe`
: Verzeichnis zur Speicherung der von den Studenten abgegebenen Lösungen. Das Unterverzeichniss für das jeweilige Übungsblatt wird automatisch angelegt, sobald der erste Student eine Lösung für das Übungsblatt abgibt.

`bin`
: Verzeichnis für die *nicht* CGI-Programme der Web-Schnittstelle.

`cgibin`
: Verzeichnis für die CGI-Programme der Web-Schnittstelle.

`preload`

Verzeichnis für die SML-Programme, die vor den Lösungen der Studenten in SML geladen werden müssen. Für jedes Übungsblatt muss hierunter ein Unterverzeichnis angelegt werden (siehe Abschnitt 3.2.1.2).

❷ An dieser Stelle werden die Vorlagendateien spezifiziert.

`anmeldeformular`

Vorlage für das Anmeldeformular.

`anmeldung`

Vorlage für die Rückmeldung des Anmeldeformulars.

`abgabeformular`

Vorlage für das Abgabeformular.

`abgabe`

Vorlage für die Rückmeldung des Abgabeformular.

`keineabgabe`

Vorlage für die Meldung bei versuchter Abgabe, wenn zur Zeit keine Abgabe möglich ist. Diese Seite kann sowohl bei Aufruf des Formulars als auch bei dem Versuch das Formular abzuschicken erscheinen, falls zwischenzeitlich der Abgabezeitraum verstrichen ist.

`abgabefehler`

Vorlage für die Rückmeldung des Abgabeformulars, falls bei der Abgabe ein Fehler aufgetreten ist.

`abgabeok`

Vorlage für die Rückmeldung des Abgabeformulars, falls die Abgabe erfolgreich durchgeführt wurde.

`belegung`

Vorlage für die Belegungsliste der Übungsgruppe.

`matrikelnummer`

Vorlage für die Rückmeldung des Formulars zum Nachtragen der Matrikelnummer.

`fehlendekonfiguration`

Vorlage für die Meldung, dass eine benötigte Konfigurationsdateien für ein Übungsblatt fehlt. Rückmeldung des Abgabeformulars.

`korrekturfehler`

Rückmeldung des Formulars zur Abgabe der korrigierten Lösungen durch die Korrektoren: Meldung, dass bei der Rückgabe ein Fehler aufgetreten ist.

`korrekturok`

Rückmeldung des Formulars zur Abgabe der korrigierten Lösungen durch die Korrektoren: Meldung über erfolgreiche Rückgabe.

`tausch`

Rückmeldung des Formulars zum Tausch von Übungsgruppen.

❸ XML-Konfigurationsdateien:

`uebungsblaetter`

Konfigurationsdatei, die die Abgabezeiträume der Übungsblätter angibt (siehe Abschnitt A.1.1.2).

`uebungsblatt_muster`
`uebungsblatt_muster_ersetzen`

Gibt die Konfigurationsdatei für ein bestimmtes Übungsblatt an. Das Vorkommen der Zeichenkette uebungsblatt_muster_ersetzen in uebungsblatt_muster wird dabei durch die zweistellige Übungsblattnummer ersetzt

❹ Spezifikation sonstiger Konfigurationswerte:

`db`

Konfiguration für den Datenbank-Zugriff

`benutzer`

Benutzer-Name für Datenbank-Zugriff der Web-Schnittstelle

`passwort`

Passwort des Benutzers für den Datenbank-Zugriff

`name`

Name der Datenbank

`belegungsfaktor`

Sobald in einer Übungsgruppe der hier angegebene prozentuale Belegungsfaktor überschritten wird, wird bei jeder Anmeldung für diese Übungsgruppe an die bei admin_email angegebene E-Mail Adresse eine Nachricht versendet.

`sml_bin`

Kommando zum Aufruf von SML.

`perl_bin`

Kommando zum Aufruf von Perl.

`prefix_preload`

Prefix der Preload-Dateien im unter preload angegebenen Verzeichnis.

`url_home`

Hauptseite der Veranstaltung. Im Kopf aller dynamisch generierten Seiten erscheint diese Adresse als Link.

`admin_email`

Von dieser E-Mail Adresse werden alle automatisch erzeugten E-Mails verschickt.

Automatische Benachrichtigungen des Systems an den Administrator werden an diese Adresse verschickt.

`veranstaltung`

Name der Veranstaltung, für die das System eingerichtet ist. Der Name erscheint im Betreff aller ausgehenden E-Mails als auch in allen dynamisch generierten Seiten.

`gruss`

Grußformel der E-Mails

A.1.1.2. `conf/xml/uebungsblaetter.xml`

A.1.1.2.1. Aufgabe

Definition der Abgabezeiträume für die Übungsblätter

A.1.1.2.2. Beispiel und Erläuterung

Beispiel A-2. Datei `conf/xml/uebungsblaetter.xml`

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>

<!DOCTYPE uebungsblaetter [                                      ❶
   <!ELEMENT uebungsblaetter (uebungsblatt)*  >
   <!ELEMENT uebungsblatt    (nummer, abgabe) >
   <!ELEMENT nummer          (#PCDATA)        >
   <!ELEMENT abgabe          (von, bis)       >
   <!ELEMENT von             (#PCDATA)        >
   <!ELEMENT bis             (#PCDATA)        >
]>

<uebungsblaetter>                                                ❷
  <uebungsblatt>
    <nummer>01</nummer>
    <abgabe>
      <von>2000-10-25-09-00</von>
      <bis>2000-10-26-17-00</bis>
    </abgabe>
  </uebungsblatt>
[...]
  <uebungsblatt>
    <nummer>11</nummer>
    <abgabe>
      <von>2001-01-18-19-00</von>
      <bis>2001-01-25-17-00</bis>
    </abgabe>
  </uebungsblatt>
</uebungsblaetter>
```

❶ DTD der Konfigurations-Datei

❷ Jedes Übungsblatt erhält in dieser Datei einen eigenen Eintrag, der den Abgabezeitraum des Übungsblatts angibt. Der Eintrag für jedes Übungsblatt ist dabei in einem eigenen uebungsblatt Element. Dieses Element beinhaltet die beiden Elemente nummer und abgabe. Inhalt des Elements nummer ist die zweistellige Nummer des Übungsblattes. Das Element abgabe hat als

Kind-Elemente von und bis. In diesen Elementen wird der Abgabezeitraum im Format CCYY-MM-DD-hh-mm-ss angegeben. Die Abgabezeiträume zweier oder mehr Übungsblätter dürfen sich dabei Überschneiden. Bei Aufruf des Abgabeformulars wird jeweils das am nächsten abzugebende Übungsblatt angezeigt, am Ende der Seite befinden sich Links auf die Abgabeseiten der übrigen zu diesem Zeitpunkt abgebbaren Übungsblätter, geordnet nach nächstem Abgabeschluss.

A.1.1.3. `conf/xml/uebungsblatt??.xml`

A.1.1.3.1. Aufgabe

Beschreibung des Übungsblatts mit der Nummer ??

A.1.1.3.2. Beispiel und Erläuterung

Beispiel A-3. Datei `conf/xml/uebungsblatt09.xml`

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>

<!DOCTYPE uebungsblatt [                                    ❶
   <!ELEMENT uebungsblatt    (nummer, information?, aufgaben) >
   <!ELEMENT nummer          (#PCDATA)                        >
   <!ELEMENT information     (#PCDATA)                        >
   <!ELEMENT aufgaben        (aufgabe)*                       >
   <!ELEMENT aufgabe         (nummer, thema, datei+)          >
   <!ELEMENT thema           (#PCDATA)                        >
   <!ELEMENT datei           (name, punkte, beschreibung,
                              vergleich?)                     >
   <!ELEMENT name            (#PCDATA)                        >
   <!ELEMENT punkte          (#PCDATA)                        >
   <!ELEMENT beschreibung    (#PCDATA)                        >
   <!ELEMENT vergleich       (#PCDATA)                        >
]>

<uebungsblatt>
  <nummer>09</nummer>                                       ❷
  <information>                                             ❸
<![CDATA[
  <p>
```

```
      Frohe Festtage und einen guten Rutsch ins neue Jahr w&uuml;nscht
      <br />
      Ihr Info1-Team
    </p>
  ]]>
    </information>

    <aufgaben>                                                 ❹

      <aufgabe>                                                ❺
        <nummer>1</nummer> <thema><![CDATA[Pr&auml;fix/Infix/Postfix]]></thema>
        <datei>                                                ❻
          <name>1.txt</name> <punkte>10</punkte>
          <beschreibung>
            <![CDATA[<tt>infix_collect A; postfix_collect A</tt>]]>
          </beschreibung>
          <vergleich>1</vergleich>
        </datei>
      </aufgabe>

  [...]

      <aufgabe>
        <nummer>4</nummer> <thema><![CDATA[Tiefendurchlauf></thema>
        <datei>
          <name>4.txt</name> <punkte>4</punkte>
          <beschreibung>
            <![CDATA[Was leistet <tt>f B</tt>?]]>
          </beschreibung>
          <vergleich>1</vergleich>
        </datei>
      </aufgabe>

    </aufgaben>
  </uebungsblatt>
```

❶ DTD der Konfigurations-Datei

❷ Als erstes muss in der Konfigurationsdatei für die Übungsblätter die jeweilige Übungsblatt-Nummer angegeben werden.

❸ Optional kann die Konfigurationsdatei zusätzliche Informationen zu dem Übungsblatt beinhalten. Diese zusätzliche Information wird im oberen Teil der generierten HTML-Datei angezeigt.

> **Wichtig:** Der Inhalt des Elements information muss, sobald ein Sonderzeichen (z.B. "ü"), ein Entity (z.B. "ü") oder ein HTML-Tag (z.B. "<p>") vorkommt durch CDATA eingeschlossen werden.

Das gleiche gilt auch für alle anderen Elemente mit PCDATA Inhalt (insbesondere die Elemente thema und beschreibung).

❹ Unterhalb dieses Elements werden die Aufgaben des Übungsblatts beschrieben. Die Anzahl der Aufgaben ist nicht begrenzt.

❺ Jede Aufgabe wird durch ein aufgabe Element beschrieben. Angegeben werden muss dabei die Nummer der Aufgabe (Element nummer), das Thema der Aufgabe (Element thema) sowie eine oder mehrere Dateien (Element datei) zu der Aufgabe.

❻ Jede Aufgabe besteht aus einer oder mehrerer Dateien. Jede Datei braucht einen für das Übungsblatt eindeutigen Dateinamen (Element name). Die maximal erreichbare Punktzahl für diese Datei muss im Element punkte angegeben werden. Dabei sind nur ganzzahlige Punktzahlen erlaubt. Im Element beschreibung muss eine kurze Beschreibung der Aufgabe, die mit der Datei gelöst werden soll, angegeben werden.

Optional kann das Element vergleich mit der Stufe des Vergleichs angegeben werden. Damit werden die abgegebenen Dateien bei der automatischen Verarbeitung miteinander Verglichen und bei Übereinstimmung mit einem entsprechenden Kommentar ergänzt.

A.1.2. Anmeldung der Studenten für die Übungsgruppen

Die Studenten melden sich elektronisch für die Übungsgruppen über ein HTML-Formular an. Ihre Anmeldedaten werden dabei in einer Datenbank gespeichert.

A.1.2.1. `www/cgi-bin/anmeldeformular.pl`

A.1.2.1.1. Aufgabe

Erzeugung des Anmeldeformulars für die Übungen (Abbildung 1-1).

A.1.2.1.2. Ablauf

Aus der Datenbank werden die Information zu allen Übungsgruppen ausgelesen. Sind in einer Übungsgruppe noch Plätze frei, so wird ein Radio-Button zur Auswahl dieser Übungsgruppe in das generierte HTML-Dokument eingefügt.

Falls mindestens eine Übungsgruppe noch nicht voll belegt ist, wird ein Submit-Button zur Anmeldung in das Dokument eingefügt.

A.1.2.1.3. Verwendete Vorlagen

`conf/tpl/anmeldeformular.html`

Vorlage für das Anmeldeformular

A.1.2.1.4. Verwendete Module

`HTML::Template`

dynamische Erzeugung der HTML-Seiten

`InfoTool::DBHelper`

Zugriff auf die Datenbank um folgende Informationen zu lesen:

- Übungsgruppen aus Tabelle uebungsgruppe
- Übungsgruppen-Leiter aus Tabelle korrektor
- bereits angemeldete Studenten aus Tabelle student

`Konfiguration` (Web)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_vorlage`

Verzeichnis mit den Vorlage-Dateien

`datei_vorlage_anmeldeformular`

Dateiname der Vorlage für das Anmeldeformular

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_url_home`
`konfiguration_veranstaltung`

Benötigt für die dynamische Erzeugung der HTML-Seiten.

A.1.2.2. `www/cgi-bin/anmeldung.pl`

A.1.2.2.1. Aufgabe

Verarbeitung der Daten von `www/cgi-bin/anmeldeformular.pl`, Eintragung in die Datenbank

A.1.2.2.2. Ablauf

1. Einfache Plausibilitätsprüfung der Angaben im Formular.
   a. Das Formular ist korrekt aufgabaut, d.h. alle Felder sind vorhanden.
   b. Alle Felder sind ausgefüllt
   c. Überprüfung auf gültige Matrikelnummer bzw. Geburtsdatum
2. Überprüfung der Benutzerkennung und des Passworts
3. Anmeldung ausführen, dabei Bearbeitung der Fälle: Neuanmeldung, Änderung der Anmeldung mit und ohne Wechsel der Übungsgruppe (Wechsel der Übungsgruppe nur erlaubt, falls der Student noch kein Übungsblatt abgegeben hat und die Konstante `$gruppe_aendern_erlaubt` - kann ganz oben in der Datei gesetzt werden - wahr ist). Der Belegungsgrad der Übungsgruppen wird dabei nochmals überprüft und eine Sperre auf der Datenbank ausgeführt, damit sichergestellt ist, dass sich in keiner Übungsgruppe mehr als die erlaubte Anzahl an Studenten anmelden kann. Meldet sich ein Student für eine Gruppe an, in der damit der Schwellenwert für die Belegung überschritten ist, wird an den Administrator eine E-Mail verschickt.
4. Rückmeldung an den Studenten, per HTML-Seite und im Erfolgsfall auch per E-Mail

A.1.2.2.3. Verwendete Vorlagen

`conf/tpl/anmeldung.html`

Vorlage für Rückmeldung über (nicht) erfolgreiche Anmeldung

A.1.2.2.4. Verwendete Module

`CGI`

Entgegennahme der Formulardaten

`HTML::Template`

dynamische Erzeugung der HTML-Seiten

`MIME::Lite`

- Versendung der E-Mail zur Bestätigung der Anmeldung an die Studenten
- Versendung der E-Mail an Administrator bei überschreiten des Schwellenwerts für die Belegung einer Übungsgruppe

`InfoTool::CheckPw`

Überprüfung der CIP-Pool Kennung und des dazugehörigen Passworts

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen und Schreiben folgender Informationen

- Daten des Studenten, Tabelle student
- Daten über tatsächliche und maximale Belegung der Übungsgruppen, Tabelle student, uebungsgruppe

`Konfiguration` (Web)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_vorlage`

Verzeichnis mit den Vorlage-Dateien

`datei_vorlage_anmeldung`

Dateiname der Vorlage für Ergebnismeldung an den Browser

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_gruss`
`konfiguration_veranstaltung`

Absender-Adresse, Gruß und Veranstaltung (für Betreff) zum Versenden der E-Mails

`konfiguration_admin_email`
`konfiguration_url_home`
`konfiguration_veranstaltung`

Benötigt für die dynamische Erzeugung der HTML-Seiten.

`konfiguration_belegungsfaktor`

Schwellenwert für Belegung der Übungsgruppen, ab der eine Warnung an die Administrator E-Mail Adresse versandt wird

A.1.2.3. `www/cgi-bin/matrikelnummer.pl`

A.1.2.3.1. Aufgabe

Verarbeitung der Daten aus dem Formular www/matrikelnummerformular.html. Dieses Formular wird von den Studenten zum Nachtrag der Matrikelnummern nach bereits erfolgter Anmeldung für eine Übungsgruppe verwendet. Dieses Formular ist notwendig, da zu Beginn des Semesters noch nicht alle Studenten ihre Matrikelnummer bekommen haben.

A.1.2.3.2. Ablauf

1. Einfache Plausibilitätsprüfung der Angaben im Formular.
    a. Das Formular ist korrekt aufgabaut, d.h. alle Felder sind vorhanden.
    b. Alle Felder sind ausgefüllt
    c. Überprüfung auf gültige Matrikelnummer bzw. Geburtsdatum
2. Überprüfung der Benutzerkennung und des Passworts
3. Matrikelnummer in der Datenbank nachtragen. Dabei Überprüfung, ob Student bereits für die Veranstaltung angemeldet ist.
4. Rückmeldung an den Studenten, per HTML-Seite und im Erfolgsfall auch per E-Mail

A.1.2.3.3. Verwendete Vorlagen

`conf/tpl/matrikelnummer.html`

Vorlage für Rückmeldung über (nicht) erfolgreichen Nachtrag der Matrikelnummer

A.1.2.3.4. Verwendete Module

`CGI`

Entgegennahme der Formulardaten

`HTML::Template`

dynamische Erzeugung der HTML-Seiten

`MIME::Lite`

- versenden der E-Mail zur Bestätigung des Nachtrags der Matrikelnummer an die Studenten

`InfoTool::CheckPw`

Überprüfung der CIP-Pool Kennung und des dazugehörigen Passworts

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen und Schreiben folgender Informationen:

- Daten des Studenten, Tabelle student

`Konfiguration` (Web)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_vorlage`

Verzeichnis mit den Vorlage-Dateien

`datei_vorlage_matrikelnummer`

Dateiname der Vorlage für Ergebnismeldung an den Browser

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_gruss`
`konfiguration_veranstaltung`

Absender-Adresse, Gruß und Veranstaltung (für Betreff) zum Versenden der E-Mails

`konfiguration_admin_email`
`konfiguration_url_home`
`konfiguration_veranstaltung`

Benötigt für die dynamische Erzeugung der HTML-Seiten.

A.1.2.4. `www/cgi-bin/tausch.pl`

A.1.2.4.1. Aufgabe

Verarbeitung der Daten aus dem Formular www/tauschformular.html. Dieses Formular dient zum Tausch von Übungsgruppen zwischen zwei Studenten mit einem Tausch-Kennwort.

Die Studenten sollen normalwerweise die Übungsgruppen nicht tauschen können. Ein Tausch soll nur nach vorheriger Genehmigung möglich sein. Bei der Genehmigung erhalten die Studenten ein Tausch-Kennwort, welches zum einmaligen Tausch von Übungsgruppen zwischen zwei Studenten berechtigt.

A.1.2.4.2. Ablauf

1. Einfache Plausibilitätsprüfung der Angaben im Formular
    a. Das Formular ist korrekt aufgebaut, d.h. alle Felder sind vorhanden
    b. Alle Felder sind ausgefüllt
2. Überprüfung der Benutzerkennung und des Passworts beider Studenten
3. Durchführung des Tauschs
    a. Überprüfung des Tausch-Kennworts
    b. Bei korrektem Kennwort: Tausch-Kennwort für nochmalige Benutzung sperren, Durchführung des Tauschs

4. Rückmeldung an die Studenten per HTML-Seite und im Erfolgsfall auch per E-Mail

A.1.2.4.3. Verwendete Vorlagen

`conf/tpl/tausch.html`
: Vorlage für Rückmeldung über (nicht) erfolgreichen Tausch der Übungsgruppen

A.1.2.4.4. Verwendete Module

`CGI`
: Entgegennahme der Formulardaten

`HTML::Template`
: dynamische Erzeugung der HTML-Seiten

`MIME::Lite`
: Versenden der E-Mail zur Bestätigung des Tauschs der Übungsgruppen an die Studenten

`InfoTool::CheckPw`
: Überprüfung der CIP-Pool Kennung und des dazugehörigen Passworts

`InfoTool::DBHelper`
: Zugriff auf die Datenbank zum Lesen und Ändern folgender Informationen:

- Daten der Studenten, insbesondere der Übungsgruppe, Tabelle student
- Tauschkennwort, Tabelle tausch

`Konfiguration` (Web)

`pfad_webbasis`
: Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_vorlage`
: Verzeichnis mit den Vorlage-Dateien

`datei_vorlage_tausch`
: Dateiname der Vorlage für Ergebnismeldung an den Browser

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_gruss`
`konfiguration_veranstaltung`

Absender-Adresse, Gruß und Veranstaltung (für Betreff) zum Versenden der E-Mails

`konfiguration_admin_email`
`konfiguration_url_home`
`konfiguration_veranstaltung`

Benötigt für die dynamische Erzeugung der HTML-Seiten.

A.1.2.5. `www/cgi-bin/belegung.pl`

A.1.2.5.1. Aufgabe

Erzeugt eine Übersichtsseite mit der Belegung der Übungsgruppen bestehend aus einer Liste aller Übungsgruppen mit ihrer aktuellen Belegung und einer alphabetisch sortierte Liste aller Studenten mit ihren Übungsgruppen.

A.1.2.5.2. Ablauf

1. Abfrage der benötigten Informationen aus der Datenbank
2. Aufbereitung und Ausgabe der Information als HTML-Seite

A.1.2.5.3. Verwendete Vorlagen

`conf/tpl/belegung.html`

Vorlage für die Belegungsliste der Übungsgruppen

A.1.2.5.4. Verwendete Module

`HTML::Template`

dynamische Erzeugung der HTML-Seiten

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen folgender Informationen:

- Daten der Übungsgruppen, Tabelle uebungsgruppe
- Daten der Studenten, Tabelle student
- Daten der Übungsgruppen-Leiter, Tabelle korrektor

`Konfiguration` (Web)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_vorlage`

Verzeichnis mit den Vorlage-Dateien

`datei_vorlage_belegung`

Dateiname der Vorlage für Belegungsliste

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_url_home`
`konfiguration_veranstaltung`

Benötigt für die dynamische Erzeugung der HTML-Seiten.

A.1.3. Abgabe der Übungsaufgaben

Die Übungsaufgaben werden elektronisch über eine HTML-Formular abgegeben. Dieses HTML-Formular wird bei jedem Aufruf für das jeweils aktuelle Übungsblatt dynamisch erzeugt.

A.1.3.1. `www/cgi-bin/abgabeformular.pl`

A.1.3.1.1. Aufgabe

Erzeugt das Abgabeformular für das zum nächsten Zeitpunkt abzugebende Übungsblatt. Optional werden auch die Abgabeformular für weitere Übungsblätter erzeugt. In beiden Fällen wird dabei die Gültigkeit des aktuellen Zeitpunkts für die Abgabe des Übungsblattes überprüft.

A.1.3.1.2. Ablauf

1. Einlesen der Datei uebungsblaetter.xml zur Bestimmung der aktuell abzugebenden Übungsblätter
2. Falls zur Zeit keine Abgabe möglich ist: Rückmeldung per HTML-Seite, dass keine Abgabe möglich ist mit Hinweis auf nächste Abgabemöglichkeit falls vorhanden

   Falls Abgabe möglich ist:

   a. Einlesen der Konfigurationsdatei für das aktuelle Übungsblatt
   b. Erzeugung der HTML-Seite für das aktuelle Übungsblatt

A.1.3.1.3. Verwendete Vorlagen

`conf/tpl/abgabeformular.html`

Vorlage für das Abgabeformular

`conf/tpl/keineabgabe.html`

Vorlage für die Rückmeldung, dass zur Zeit keine Abgabe möglich ist

`conf/tpl/fehlendekonfiguration.html`

Vorlage für die Rückmeldung, dass die Konfigurationsdatei für das aktuelle Übungsblatt fehlt

A.1.3.1.4. Verwendete Konfigurationsdateien

`conf/xml/uebungsblaetter.xml`

Abgabezeiträume für die einzelnen Übungsblätter

`conf/xml/uebungsblatt??.xml`

Konfiguration für ein Übungsblatt

A.1.3.1.5. Verwendete Module

`HTML::Template`
: dynamische Erzeugung der HTML-Seiten

`InfoTool::Datum`
: Datumsberechnung und -formatierung

`InfoTool::ParserUebungsblaetter`
: Ermittlung des aktuellen Übungsblattes anhand der in der Datei `conf/xml/uebungsblaetter.xml` angegebenen Abgabezeiträume

`InfoTool::ParserUebungsblatt`
: Auslesen von Informationen zum aktuellen Übungsblatt aus der Datei `conf/xml/uebungsblatt??.xml`

`Konfiguration` (Web)

- `pfad_webbasis`
  Basispfad für alle anderen Pfade der Web-Schnittstelle
- `pfad_konfiguration`
  Verzeichnis für die Konfigurationsdateien
- `pfad_vorlage`
  Verzeichnis mit den Vorlage-Dateien
- `datei_vorlage_abgabeformular`
  Dateiname der Vorlage für das Abgabeformular
- `datei_vorlage_fehlendekonfiguration`
  Dateiname der Vorlage für die Meldung, dass die Konfigurationsdatei für das Übungsblat fehlt
- `datei_vorlage_keineabgabe`
  Dateiname der Vorlage für die Meldung, dass zur Zeit keine Abgabe von Übungsblättern möglich ist
- `datei_konfiguration_uebungsblaetter`
  Dateiname der Konfigurationsdatei für Abgabezeiträume der Übungsblätter

`datei_konfiguration_uebungsblatt_muster`
`datei_konfiguration_uebungsblatt_muster_ersetzen`

Dateiname der Konfigurationsdatei für das Übungsblatt

`konfiguration_admin_email`
`konfiguration_url_home`
`konfiguration_veranstaltung`

Benötigt für die dynamische Erzeugung der HTML-Seiten.

A.1.3.2. `www/cgi-bin/abgabe.pl`

A.1.3.2.1. Aufgabe

Entegegennahme und Verarbeitung der Daten aus `www/cgi-bin/abgabeformular.pl`

A.1.3.2.2. Ablauf

Im folgenden der Ablauf bei Abgabe des Übungsblattes X:

1. Datei uebungsblaetter.xml zur Bestimmung der aktuell abzugebenden Übungsblätter einlesen
2. falls zur Zeit keine Abgabe von Übungsblättern möglich ist: Rückmeldung per HTML-Seite, dass keine Abgabe möglich ist mit Hinweis auf nächste Abgabemöglichkeit für ein Übungsblatt (falls vorhanden)

   falls Abgabe eines Übungsblattes möglich ist (unabhängig davon, ob es für Übungsblatt X möglich ist):

   a. einfache Plausibilitätsprüfung der Angaben im Formular
      i. Formular ist korrekt aufgebaut, d.h. alle Felder sind vorhanden
      ii. alle Felder sind ausgefüllt
   b. Überprüfen, ob Übungsblatt X zur Zeit abgegeben werden kann
   c. Benutzerkennung und des Passworts des abgebenden Studenten überprüfen
   d. Überprüfung der abzugebenden Dateien:
      i. Konfigurationsdatei für Übungsblatt X einlesen
      ii. für jede abzugebenden Dateie:

A. Datei wird (nichtleer) abgegeben oder es ist für die Datei explizit angegeben, dass sie nicht abgegeben werden soll

B. Datei einlesen

C. Überprüfen des Dateityps mit **file** (nur Dateityp „text“ wird angenommen)

D. Umwandlung von DOS (\r\n) und Mac (\r) Zeilenenden nach UNIX (\n)

E. falls Preload-Datei vorhanden: Einlesen und abgegebene Datei an Preload Datei anhängen

F. falls SML-Datei: Syntaxcheck (mit `KommSML`)

e. Anmeldung aller Teammitglieder überprüfen

f. Überprüfen, ob alle Temmitglieder beim gleichen Korrektor sind

g. Überprüfen, dass keines der Teammitglieder bereits eine Lösung für dieses Übungsblatt abgegeben hat

dabei Datenbank für Schreibzugriffe auf Tabelle abgabe sperren

h. abgegebene Dateien in Abgabe-Verzeichnis speichern

erfolgreiches Speichern überprüfen

i. Abgabe in Datenbank eintragen

Sperre der Datenbank aufheben

j. E-Mails über erfolgreiche Abgabe an alle Teammmitglieder versenden

k. Seite über erfolgreiche Abgabe anzeigen

A.1.3.2.3. Verwendete Vorlagen

`conf/tpl/abgabeok.html`

Vorlage für die Rückmeldung über eine erfolgreiche Abgabe

`conf/tpl/abgabefehler.html`

Vorlage für die Rückmeldung über einen Fehler bei der Abgabe

`conf/tpl/keineabgabe.html`

Vorlage über die Rückmeldung, dass zur Zeit keine Abgabe möglich ist

A.1.3.2.4. Verwendete Konfigurationsdateien

`conf/xml/uebungsblaetter.xml`

Abgabezeiträume für die einzelnen Übungsblätter

`conf/xml/uebungsblatt??.xml`

Konfiguration für ein Übungsblatt

A.1.3.2.5. Verwendete Module

`CGI`

Entgegennahme der Formulardaten

`HTML::Template`

dynamische Erzeugung der HTML-Seiten

`IO::File`

Erzeugung von temporären Dateien für Typüberprüfung mit **file** (zusammen mit `POSIX qw(tmpnam)`)

`MIME::Lite`

Bestätigung der erfolgreichen Abgabe per E-Mail an alle Teammitglieder.

`POSIX qw(tmpnam)`

Erzeugung von temporären Dateien für Typüberprüfung mit **file** (zusammen mit `IO::File`)

`InfoTool::CheckPw`

Überprüfung der CIP-Pool Kennung und des dazugehörigen Passworts

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen und Schreiben folgender Informationen:

- Daten über Abgaben, Tabelle abgabe
- Daten über Studenten, Tabelle student

`InfoTool::KommSML`

Kommunikation mit SML

`InfoTool::ParserUebungsblaetter`

Ermittlung des aktuellen Übungsblattes anhand der in der Datei `conf/xml/uebungsblaetter.xml` angegebenen Abgabezeiträume

`InfoTool::ParserUebungsblatt`

Auslesen von Informationen zum aktuellen Übungsblatt aus der Datei `conf/xml/uebungsblatt??.xml`

`Konfiguration` (Web)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_abgabe`

Verzeichnis, unter dem die entgegengenommenen Übungsaufgaben, nach Übungsblatt und Teammitgliedern sortiert, abgespeichert werden

`pfad_konfiguration`

Verzeichnis für die Konfigurationsdateien

`pfad_vorlage`

Verzeichnis mit den Vorlage-Dateien

`datei_vorlage_abgabefehler`

Dateiname der Vorlage für die Meldung, dass die Abgabe aufgrund eines Eingabefehlers nicht angenommen wurde

`datei_vorlage_keineabgabe`

Dateiname der Vorlage für die Meldung, dass zur Zeit keine Abgabe von Übungsblättern möglich ist

`datei_vorlage_ok`

Dateiname der Vorlage für die Meldung, das die Abgabe erfolgreich verlaufen ist

`datei_konfiguration_uebungsblaetter`

Dateiname der Konfigurationsdatei für Abgabezeiträume der Übungsblätter

`datei_konfiguration_uebungsblatt_muster`
`datei_konfiguration_uebungsblatt_muster_ersetzen`

Dateiname der Konfigurationsdatei für das Übungsblatt

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_gruss`
`konfiguration_veranstaltung`

Absender-Adresse, Gruß und Veranstaltung (für Betreff) zum Versenden der E-Mails

`konfiguration_admin_email`
`konfiguration_url_home`
`konfiguration_veranstaltung`

Benötigt für die dynamische Erzeugung der HTML-Seiten.

`konfiguration_sml_bin`

Kommando zum Starten von SML

A.1.4. Rückgabe der korrigierten Lösungen

Die Korrektoren können die korrigierten Übungsaufgaben über die Web-Schnittstelle abgeben. Die Programme für die Rückgabe bei Korrektur im Korrektoren-Verzeichnis beschreibt Abschnitt A.3.3.

A.1.4.1. `www/cgi-bin/korrektur.pl`

A.1.4.1.1. Aufgabe

Web-Frontend für `bin/bewertungen.pl`. Nimmt die korrigierten Übungsaufgaben über das Formular `www/korrekturformular.html`entgegen, ruft `bin/bewertungen.pl` auf und archiviert die korrigierten Aufgaben im Erfolgsfall.

A.1.4.1.2. Ablauf

1. Einfache Plausibilitätsprüfung der Angaben im Formular.
    a. Das Formular ist korrekt aufgabaut, d.h. alle Felder sind vorhanden.
    b. Alle Felder sind ausgefüllt
2. Überprüfung der Benutzerkennung und des Passworts
3. Archiv in `pfad_korrektur_arbeit`, Unterverzeichnis für Korrektor, speichern und entpacken (mit **tar xfz** oder **unzip -U**)
4. Überprüfung, ob Benutzerkennung die eines Korrektors ist

5. Rücknahme der korrigierten Aufgaben durchführen:
    a. **bewertungen.pl** aufrufen
    b. korrigierte Aufgaben unter `pfad_korrektur_archiv`, Unterverzecihnis für Korrektor, archivieren
6. Seite über erfolgreiche Rücknahme der korrigierten Lösungen zeigen

A.1.4.1.3. Verwendete Vorlagen

`conf/tpl/korrekturok.html`

Vorlage für Rückmeldung über erfolgreiche Rücknahme der korrigierten Übungsaufgaben

`conf/tpl/korrekturfehler.html`

Vorlage für Rückmeldung über Fehler bei der Rücknahme der korrigierten Übungsaufgaben

A.1.4.1.4. Verwendete Module

`CGI`

Entgegennahme der Formulardaten

`HTML::Template`

dynamische Erzeugung der HTML-Seiten

`InfoTool::CheckPw`

Überprüfung der CIP-Pool Kennung und des dazugehörigen Passworts

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen und Schreiben folgender Informationen:

- Daten der Korrektoren, Tabelle korrektor

`Konfiguration` (Web)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_bin`

Verzeichnis, in dem sich **bewertungen.pl** und **punktestand.pl** befinden

`pfad_korrektur`
`pfad_korrektur_arbeit`
`pfad_korrektur_archiv`

Verzeichnis zur temporären Lagerung der korrigierten Aufgaben zur Weiterverarbeitung (`pfad_korrektur_arbeit`)

Verzeichnis zur Archivierung der korrigierten Aufgaben (`pfad_korrektur_archiv`)

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_url_home`
`konfiguration_veranstaltung`

Benötigt für die dynamische Erzeugung der HTML-Seiten.

A.1.4.2. `bin/bewertungen.pl`

Wird von `www/cgi-bin/korrektur.pl` aufgerufen.

A.1.4.2.1. Aufgabe

Erfasst die Bewertungen der Korrektoren. Die Punktzahlen aus den korrigierten Aufgaben werden in die Datenbank eingetragen und an die Teammitglieder verschickt. Die Teammitglieder erhalten zusätzlich die kommentierten Lösungen ihrer Aufgaben zugeschickt.

Die Korrektoren und Übungsgruppen-Leiter erhalten die aktuellen Punktestände ihrer Übungsgruppen per E-Mail zugeschickt.

Siehe auch Abschnitt A.3.3.2.

A.1.4.2.2. Ablauf

1. Überprüfung der Kommandozeilenparameter
2. Einlesen aller korrigierten Lösungen, Einlesen der Informationen aus dem Kopf der Lösungen und Aufbau einer Perl-Datenstruktur mit den Informationen (Kennung des Studenten, Aufgabe und erreichte Punkte) aus diesen Lösungen

dabei Überprüfung, ob bei jeder Lösung eine erreichte Punktzahl eingetragen wurde und ob diese Punktzahl nicht über der maximal erreichbaren Punktzahl liegt

3. Eintrag dieser Informationen in die Datenbank
4. erstellen eines Archivs der Lösungen für jedes Team und verschicken dieses Archivs per E-Mail an alle Team-Mitglieder
5. Ermitteln der korrigierten Übungsgruppen
6. Versenden der aktuellen Punktestände der korrigierten Übungsgruppen an den Korrektor und den Übungsgruppen-Leiter

A.1.4.2.3. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`MIME::Lite`

- Versenden der bewerteten und korrigierten Aufgaben per E-Mail an die Studenten
- Versenden der aktuellen Punktestände per E-Mail an den Korrektor und den Übungsgruppen-Leiter

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen und Schreiben folgender Informationen:

- Daten des Studenten, Tabelle student
- Punkte der Studenten für die Aufgaben, Tabelle punkte und aufgabe
- Leiter und Korrektoren der Übungsgruppen, Tabelle uebungsgruppe und korrektor

`Konfiguration` (Web)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_bin`

Verzeichnis, in dem sich **bewertungen.pl** und **punktestand.pl** befinden `pfad_cgibin`

Verzeichnis, in dem sich `Konfiguration.pm` befindet

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_gruss`
`konfiguration_veranstaltung`

Absender-Adresse, Gruß und Veranstaltung (für Betreff) zum Versenden der E-Mails

A.1.4.3. `bin/punktestand.pl`

Wird von `bin/bewertungen.pl` aufgerufen.

A.1.4.3.1. Aufgabe

Ausgabe des aktuellen Punktestand einer Übungsgruppe oder aller Übungsgruppen.

A.1.4.3.2. Ablauf

1. Überprüfung der Kommandozeilenparameter
2. Aufbau einer Perl Datenstruktur mit allen benötigten Informationen:
    a. Informationen zur Übungsgruppe (Korrektor, Spitzname)
    b. Name, Vorname der Studenten
    c. Summe der Punkte eines jeden Studenten für jedes Übungsblatt
    d. Punkte eines jeden Studenten in allen bisherigen Klausuren (optional)
3. Bestimmung der maximal möglichen Punkte für jedes Übungsblatt aufgrund der Konfigurationsdateien `conf/xml/uebungsblatt??.xml`
4. Ausgabe der Punkteliste(n)

A.1.4.3.3. Verwendete Konfigurationsdateien

`conf/xml/uebungsblatt??.xml`

Konfiguration für ein Übungsblatt

A.1.4.3.4. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen folgender Informationen:

- Daten der Studenten, Tabelle student
- Daten zu den Übungsgruppen und den zugehörigen Korrektoren, uebungsgruppe und korrekor
- erreichte Punkte der Studenten aus Tabellen aufgabe und punkte

`InfoTool::ParserUebungsblatt`

Auslesen der maximal möglichen Punktzahl für die Übungsblätter aus den Dateien `conf/xml/uebungsblatt??.xml`

`Konfiguration` (Web)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_konfiguration`

Verzeichnis für die Konfigurationsdateien

`datei_konfiguration_uebungsblatt_muster`
`datei_konfiguration_uebungsblatt_muster_ersetzen`

Dateiname der Konfigurationsdatei für das Übungsblatt

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

A.2. Administration und Organisation

Für die Administration und Organisation stehen einige Hilfsmittel zur Verfügung. Diese umfassen unter anderen Schnittstellen zur Datenbank und Überprüfungen von Konfigurationsdateien. Es sind also Hilfsmittel, die nicht notwendigerweise zum Betrieb des Systems gebraucht werden, den Umgang damit aber erleichtern.

A.2.1. Konfigurationsdateien

Zur Konfiguration kommt hier nur ein Perl Modul zum Einsatz. Es wird jedoch teilweise auch auf die XML-Konfigurationsdateien der Web-Schnittstelle zugegriffen.

A.2.1.1. `bin/Konfiguration.pm`

Zentrale Konfigurationsdatei für die Perl Programme zur Administration und Organisation.

A.2.1.1.1. Aufgabe

Konfiguration aller von der jeweiligen Installation abhängigen von den Perl Programmen benötigter Werte wie z.B. Verzeichnis- und Dateinamen.

A.2.1.1.2. Beispiel und Erläuterung

Beispiel A-4. Datei `bin/Konfiguration.pm`

```
#!/usr/bin/perl -w

##############################################################################
#Name:  Konfiguration.pm
#Projekt: Projektarbeit - Einsatz elektronischer Medien
#  für den Übungsbetrieb
#Sinn&Zweck: Konfiguration für den nicht Web-Teil
#Autor:  Georg Schneemayer
##############################################################################

use strict;

package Konfiguration;

#Pfade
#-Web                                                          ❶
$Konfiguration::pfad_webbasis = "/home/proj/info1run/";
$Konfiguration::pfad_konfiguration = "conf/xml/";
$Konfiguration::pfad_vorlage = "conf/tpl/";
```

```
$Konfiguration::pfad_abgabe = "abgabe/";
#-Tool                                                  ❷
$Konfiguration::pfad_toolbasis = "/home/proj/info1korrektur/";
$Konfiguration::pfad_archiv = "archiv/";
$Konfiguration::pfad_arbeit = "tmp/arbeit/";
$Konfiguration::pfad_korrektor = "korrektor/";
$Konfiguration::pfad_testsuites = "testsuites/";
$Konfiguration::pfad_bin = "bin/";

#Dateien
#-Vorlagen                                              ❸
$Konfiguration::datei_vorlage_abgabeformular = "abgabeformular.html";
$Konfiguration::datei_vorlage_fehlendekonfiguration = "fehlendekonfiguration.html";
#-Konfiguration                                         ❹
$Konfiguration::datei_konfiguration_uebungsblaetter = "uebungsblaetter.xml";
$Konfiguration::datei_konfiguration_uebungsblatt_muster = "uebungsblattYXC.xml";
$Konfiguration::datei_konfiguration_uebungsblatt_muster_ersetzen = "YXC";\
  # Achtung: kein Sonderzeichen von regulären Ausdrücken erlaubt!

#Konfiguration                                          ❺
$Konfiguration::konfiguration_db_benutzer = "info1tool";
$Konfiguration::konfiguration_db_passwort = "YXCVBNM";
$Konfiguration::konfiguration_db_name = "info1";
$Konfiguration::konfiguration_sml_bin =\
  "/home/proj/info1abg/bin/ulimit_restricted_bash_SML.sh";
$Konfiguration::konfiguration_prefix_test = "test-";
$Konfiguration::konfiguration_prefix_preload = "preload-";
$Konfiguration::konfiguration_url_home =\
  "http://www.pms.informatik.uni-muenchen.de/lehre/info1/00ws01/";
$Konfiguration::konfiguration_admin_email = "info1\@informatik.uni-muenchen.de";
$Konfiguration::konfiguration_veranstaltung = "Informatik I (WS 2000/2001)";
$Konfiguration::konfiguration_gruss = "Grüße,\n\n Info I-Team";
@Konfiguration::konfiguration_vergleich_bin =
  ( {"command" =>
       "diff -s --brief DATEI1 DATEI2",
     "result" =>
       "Files DATEI1 and DATEI2 are identical"},
    {"command" =>
      "diff -s --brief --ignore-all-space --ignore-blank-lines --ignore-
case --ignore-space-change DATEI1 DATEI2",
     "result" =>
       "Files DATEI1 and DATEI2 are identical"});

$Konfiguration::konfiguration_kommentar =
  {"sml" => ["\n(*\nINFO-START\n", "\n", "\nINFO-ENDE\n*)\n"],
   "xml" => ["\n<!--\nINFO-START\n", "\n", "\nINFO-ENDE\n-->\n"],
   "java"=> ["\n/*\nINFO-START\n", "\n", "\nINFO-ENDE\n*/\n"],
   "pl"  => ["\n#\n#INFO-START\n", "\n#", "\n#INFO-ENDE\n"],
   "txt" => ["\n#\n#INFO-START\n", "\n#", "\n#INFO-ENDE\n"]};
```

```
$Konfiguration::konfiguration_kommentar_vergleich =
  {"sml" => ["\n(*\nVERGLEICH-START\n", "\n", "\nVERGLEICH-ENDE\n*)\n"],
   "xml" => ["\n<!--\nVERGLEICH-START\n", "\n", "\nVERGLEICH-ENDE\n-->\n"],
   "java"=> ["\n/*\nVERGLEICH-START\n", "\n", "\nIVERGLEICH-ENDE\n*/\n"],
   "pl"  => ["\n#\n#VERGLEICH-START\n", "\n#", "\n#VERGLEICH-ENDE\n"],
   "txt" => ["\n#\n#VERGLEICH-START\n", "\n#", "\n#VERGLEICH-ENDE\n"]};


$Konfiguration::konfiguration_kommentar_autotest =
  {"sml" => ["\n(*\nTEST-START\n", "\n", "\nTEST-ENDE\n*)\n"]};
```

❶ An dieser Stelle müssen einige der Pfade der Web-Schnittstelle eingetragen werden. Erlauterung in Abschnitt A.1.1.1.

❷ An dieser Stelle müssen die Pfade zu den Dateien der Administrations-Schnittstelle eingetragen werden. Im einzelnen sind dies:

`toolbasis`

: Basis-Verzeichnis für die Administrations-Schnittstelle. Alle anderen Pfade sind relativ zu diesem Verzeichnis.

`archiv`

: Archiv Verzeichnis für die von den Studenten abgegebenen Lösungen.

`arbeit`

: Verzeichnis für temporäre Dateien während der der Aufbereitung der Lösungen zur Korrektur

`korrektor`

: Verzeichnis der Korrektoren. Unterhalb dieses Verzeichnis hat jeder der Korrektoren ein eigenes Unterverzeichnis, in dem er die von ihm zu korrigierenden Aufgaben vorfindet.

`testsuites`

: Verzeichnis mit Test-Suites und Preload-Dateien zu den abgegebenen Lösungen der Studenten. Für jedes Übungsblatt existiert hier ein eigenes Unterverzeichnis. (siehe Abschnitt 3.2.1.3 und Abschnitt 3.2.1.2)

`bin`

: Verzeichnis für die Programme der Administrations-Schnittstelle.

❸ An dieser Stelle werden die Vorlagendateien spezifiziert. Erlauterung in Abschnitt A.1.1.1.

❹ XML-Konfigurationsdateien Erlauterung in Abschnitt A.1.1.1.

❺ Spezifikation sonstiger Konfigurationswerte. Die hier nicht erläuterten Wert finden sich in Abschnitt A.1.1.1.

`db`

: Konfiguration für den Datenbank-Zugriff

  `benutzer`

  : Benutzer-Name für Datenbank-Zugriff der Werkzeuge

`prefix_test`

: Prefix der Test-Dateien im unter `testsuites` angegebenen Verzeichnis.

`vergleich_bin`

: Konfiguration der Vergleich-Stufen. Für jede Vergleich Stufe muss dabei ein Eintrag der Form

```
{"command" => "Aufruf_des_Vergleich-Programms DATEI1 DATEI2",
 "result" => "Meldung_bei_Gleichheit"}
```

hinzugefügt werden. Dabei werden `DATEI1` und `DATEI2` zur Laufzeit automatisch mit den beiden zu vergleichenden Dateien ersetzt.

`konfiguration_kommentar`
`konfiguration_kommentar_vergleich`
`konfiguration_kommentar_autotest`

: Konfiguration der Kommentarzeichen. Dabei ist `konfiguration_kommentar` die Konfiguration für die allgemeinen Kommentarzeichen zu Beginn jeder Lösung, `konfiguration_kommentar_vergleich` die Konfiguration der Kommentarzeichen der Vergleiche und `konfiguration_kommentar_autotest` die Konfiguration der Kommentarzeichen der automatischen Tests.

Das Format ist dabei wie folgt:

```
"Dateiendung" =>
        ["Kommentarzeichen und -text Start",
         "Kommentarzeichen und -text zu Beginn jeder neuen Zeile",
         "Kommentarzeichen und -text Ende"]
```

Zu beachten ist dabei, dass auch der Zeilenumbruch (\n) mitangegeben werden muss.

A.2.2. Hilfsmittel zur Konfiguration der Web-Schnittstelle

Da Änderungen an der Konfiguration der Web-Schnittstelle sofort „live“ gehen, sind Hilfsmittel zum Testen von modifizierten Konfigurationen notwendig.

A.2.2.1. `bin/test_abgabeformular`

Dieses Programm ist bis auf die fehlende Überprüfung der Gültigkeit einer Abgabe für das spezifizierte Übungsblatt zum aktuellen Zeitpunkt und der Kommandozeilen- anstatt der CGI-Schnittstelle identische mit der Datei `www/cgi-bin/abgabeformular` (siehe Abschnitt A.1.3.1).

A.2.2.1.1. Aufgabe

Gibt das Abgabeformular für das spezifizierte Übungsblatt unabhängig vom Abgabezeitraum aus.

A.2.2.1.2. Ablauf

Bis auf fehlende Überprüfung des Abgabezeitraums analog zu `www/cgi-bin/abgabeformular` (siehe Abschnitt A.1.3.1).

A.2.2.1.3. Verwendung

test_abgabeformular.pl {-u *Übungsblatt-Nummer*}

A.2.2.1.4. Verwendete Vorlagen

siehe Abschnitt A.1.3.1

A.2.2.1.5. Verwendete Module

Es werden die gleichen Module wie bei `www/cgi-bin/abgabeformular.pl` (siehe Abschnitt A.1.3.1) verwendet. Ausnahmen sind die Verwendung von `Getopt::Std` statt `CGI` und `Konfiguration` (Tool) statt `Konfiguration` (Web).

A.2.3. Schnittstelle LaTeX und Datenbank

A.2.3.1. `bin/latex-db.pl`

A.2.3.1.1. Aufgabe

Überträgt Informationen aus einer LaTeX Datei in die Datenbank. Die LaTeX Datei wird unter anderem für die Erstellung von Zulassungslisten für die Klausuren benötigt. In ihr werden auch die Klausurergebnisse eingetragen.

A.2.3.1.2. Verwendung

latex-db.pl {-f *`LaTeX-Datei`*} [-k *`Klausur`* | -p *`Punkte`*]

A.2.3.1.3. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen und Schreiben folgender Informationen:

- Ergebnisse der Studenten bei den Klausuren, Tabelle klausur_student

`Konfiguration` (Tool)

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

A.2.3.2. `bin/db-latex.pl`

A.2.3.2.1. Aufgabe

Erzeugung der LaTeX-Datei (siehe Abschnitt A.2.3.1).

A.2.3.2.2. Verwendung

db-latex.pl {-k `Klausur`} {-u `von-bis (Übungsblatt)`} [-f `Punkte (fragliche Zulassung)`] {-z `Punkte (sichere Zulassung)`} [-p]

Bei Option `-p` wird anstatt des Zulassungscodes die Punktzahl ausgegeben.

A.2.3.2.3. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen und Schreiben folgender Informationen:

- Daten des Studenten, Tabelle student
- Erreichte Punkte, Tabelle punkte und aufgabe
- Klausur-Ergebnisse, Tabelle klausur_student

`Konfiguration` (Tool)

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

A.2.4. Sonstige Hilfsmittel

A.2.4.1. `bin/nachkorrektur.pl`

A.2.4.1.1. Aufgabe

Trägt Korrekturen außerhalb des normalen Systems in die Datenbank ein. Dies können nachträgliche Bewertungsänderungen sowie auch Korrekturen von nicht über die Web-Schnittstelle abgegebenen Aufgaben sein.

Zusätzlich wird der Student per E-Mail entsprechend benachrichtigt.

A.2.4.1.2. Verwendung

nachkorrektur.pl {-k *`Student-Kennung`*} {-p *`Punkte`*} {-u *`Übungsblatt`*} {-f *`Dateiname der korrigierten Aufgabe`*}

A.2.4.1.3. Verwendete Konfigurationsdateien

`conf/xml/uebungsblatt??.xml`

Konfiguration für ein Übungsblatt

A.2.4.1.4. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`MIME::Lite`

Versenden der E-Mail zur Benachrichtigung des Studenten

`InfoTool::ParserUebungsblatt`

Auslesen der maximal möglichen Punktzahl für die Übungsblätter aus den Dateien `conf/xml/uebungsblatt??.xml` (Abschnitt A.1.1.3)

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen folgender Informationen:

- Daten der Studenten, Tabelle student
- Daten zu den Übungsgruppen und den zugehörigen Korrektoren, uebungsgruppe und korrekor
- erreichte Punkte der Studenten aus Tabellen aufgabe und punkte

`Konfiguration` (Tool)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`datei_konfiguration_uebungsblatt_muster`
`datei_konfiguration_uebungsblatt_muster_ersetzen`

Dateiname der Konfigurationsdatei für das Übungsblatt

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_gruss`
`konfiguration_veranstaltung`

Absender-Adresse, Gruß und Veranstaltung (für Betreff) zum Versenden der E-Mails

A.2.4.2. `bin/korrekturstand.pl`

A.2.4.2.1. Aufgabe

Listet Studenten und deren Korrektoren auf, für die ein bestimmtes Übungsblatt noch nicht korrigiert wurde.

A.2.4.2.2. Verwendung

korrekturstand.pl {-u *`Übungsblatt`*}

A.2.4.2.3. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen folgender Informationen:

- Daten über die Abgabe, Tabelle abgabe
- Daten über die Aufgaben eines Übungsblatts, Tabelle aufgabe
- Daten über die erreichten Punkte, Tabelle punkte
- Daten der Studenten, Tabelle student

- Daten der Korrektoren, Tabelle korrektor

`Konfiguration` (Tool)

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`
: Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

A.2.4.3. `bin/punktestand.pl`

A.2.4.3.1. Aufgabe

Gibt den aktuellen Punktestand einer oder aller Übungsgruppen aus.

Bis auf die Verwendung von `Konfiguration.pm` (Tool) anstatt `Konfiguration.pm` (Web) identisch mit punktestand.pl (Abschnitt A.1.4.3) der Web-Schnittstelle.

A.2.4.3.2. Verwendung

punktestand.pl [-g *Übungsgruppe*] [-a]

Wird Option `-a` angegeben, werden auch die Studenten mit ausgegeben, die kein einziges Übungsblatt abgegeben haben.

A.2.4.3.3. Verwendete Module

siehe Abschnitt A.1.4.3

A.2.4.4. `bin/testen.pl`

A.2.4.4.1. Aufgabe

Testet ein SML-Programm unter Angabe der SML-Datei und der Test-Datei. Dabei sind je zwei Tests voneinander unabhängig.

Die Testdatei hat dabei das Format:

Anweisung
Ergebnis
Anweisung
Ergebnis

testen.pl wird verwendet von **autotest.pl** (Abschnitt A.3.2.4)

A.2.4.4.2. Verwendung

testen.pl {-s *`SML-Programm`*} {-t *`Test-Datei`*} [-p *`Preload-Datei`*] [-d] {-f *`Dateiname der korrigierten Aufgabe`*}

Bei Option `-d` werden nur Diskrepanzen angezeigt. Erfolgreich durchgeführte Tests werden hingegen nicht ausgegeben.

A.2.4.4.3. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`InfoTool::KommSML`

Kommunikation mit SML

`Konfiguration` (Tool)

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`sml_bin`

Kommando zum Aufruf von SML.

A.2.4.5. `bin/mail-matrikelnummernachtrag.pl`

A.2.4.5.1. Aufgabe

Erinnert alle Studenten, die noch nicht ihre vollständige Matrikelnummer angegeben haben, per E-Mail daran, die Matrikelnummer nachzutragen

> **Tip:** Dieses Programm ist nur ein schneller Hack. Es kann jedoch leicht angepasst werden, um eine beliebige E-Mail an eine durch eine SQL-Abfrage spezifizierte Gruppe von Studenten zu verschicken.

A.2.4.5.2. Verwendung

mail-matrikelnummernachtrag.pl

Keine Kommandozeilenparameter Parameter.

A.2.4.5.3. Verwendete Module

`MIME::Lite`

- Versenden der E-Mails

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen folgender Informationen:

- Ermittlung der Studenten, an die die E-Mail verschickt werden soll, Tabelle student

`Konfiguration` (Tool)

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_gruss`
`konfiguration_veranstaltung`

Absender-Adresse, Gruß und Veranstaltung (für Betreff) zum Versenden der E-Mails

A.3. Verarbeitung der Lösungen

In diesem Abschnitt wird auf die Programme, die zur automatischen Verarbeitung der Lösungen dienen, eingegangen. Dies umfasst die Programme, die die Lösungen für die Korrektur aufbereiten und die Programme, die für die Rücknahme der korrigierten Lösungen zuständig sind.

A.3.1. Konfigurationsdateien

Hierfür kommen wieder die selben Konfigurationsdateien wie bei Abschnitt A.2.1, Administration und Organisation - Konfigurationsdateien, zum Einsatz. Darüber hinaus kann es für einzelne Aufgaben noch Testsuites und Preload-Dateien geben.

A.3.1.1. `testsuites` - Testsuites

Im Verzeichnis `testsuites` befinden sich die Testsuites für die einzelnen Übungsblätter. Jedes Übungsblatt hat dabei ein eigenes Unterverzeichnis (z.B. 06 für das sechste Übungsblatt). Darunter befinden sich die Testdateien. Ihr Name ist bis auf das Prefix `konfiguration_prefix_test` identisch mit den Dateinamen der zu testenden Programme.

Die einzelnen Testsuites bestehen jeweils aus mehreren Tests und diese wiederum aus je zwei Zeilen. Die erste Zeile ist die Anweisung für das zu testende Programm, die zweite die erwartete Ausgabe.

Hier ein kleiner Ausschnitt bestehend aus drei Tests einer Test-Datei:

```
eval( variable "x", [(variable "x",1), (variable "y",2), (variable "x",4), (variable "z",3)] )
1 : int
eval( constant 7, [(variable "x",1), (variable "y",2), (variable "x",4), (variable "z",3)] )
7 : int
eval( product(sum(variable "x",constant 7), variable "y"), [(variable "x",1), (variable "y",2),\
  (variable "x",4), (variable "z",3)] )
16 : int
```

A.3.1.2. `testsuites` - Preload-Dateien

Für einige Tests kann es notwendig sein, dass das abgegebene Programm, etwa um Ausgaberoutinen, erweitert wird. Dies kann mit den Preload-Dateien geschehen. Ihr Name ist bis auf den Prefix `konfiguration_prefix_preload` identisch mit der jeweiligen Datei.

Die Preload-Dateien werden bei jedem Test vor die zu testende Datei gehängt.

A.3.2. Aufbereitung für die Korrektur

Die folgenden Programme werden im Allgemeinen nicht einzeln gestartet, sondern über das bash-Skript **startTools.sh**. Die Ausführung kann je nach Anzahl der abgegebenen Aufgaben und Menge der Testsuites lange dauern (~1h).

A.3.2.1. `bin/startTools.sh`

A.3.2.1.1. Aufgabe

Startet die Aufbereitung. Ruft die folgenden Programme mit den notwendigen Parametern für das angegebene Übungsblatt auf.

A.3.2.1.2. Verwendung

startTools.sh {-u *Übungsblatt*}

A.3.2.2. `bin/archivierung.pl`

A.3.2.2.1. Aufgabe

Kopiert die abgegebenen Aufgaben aus dem Web-Verzeichnis in ein Verzeichnis zur weiteren Verarbeitung der Aufgaben (Standard: `pfad_arbeit`). Anschließend werden die abgegebenen Lösungen archiviert (Standard: `pfad_archiv`).

A.3.2.2.2. Verwendung

archivierung.pl [-g *Abgabe-Pfad*] [-c *Archiv-Pfad*] [-b *Arbeits-Pfad*] {-u *Übungsblatt*}

A.3.2.2.3. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`Konfiguration` (Tool)

`pfad_webbasis`

Basispfad für alle anderen Pfade der Web-Schnittstelle

`pfad_toolbasis`

Basispfad für alle anderen Pfade der Administrations-Schnittstelle

`abgabe_pfad`

Abgabe-Pfad, falls nicht anders angegeben

`archiv_pfad`

Archiv-Pfad, falls nicht anders angegeben

`arbeit_pfad`

Arbeits-Pfad, falls nicht anders angegeben

A.3.2.3. `bin/vergleiche.pl`

A.3.2.3.1. Aufgabe

Vergleicht die von den Studenten abgebenen Lösungen mit Hilfe externer Programme

A.3.2.3.2. Ablauf

1. Auswertung der Kommandozeile
2. Einlesen der Konfigurationsdatei für das Übungsblatt um festzustellen, welche Vergleiche durchgeführt werden sollen
3. Erstellen einer Dateiliste mit allen abgegeben Dateien und der für sie durchzuführenden Stufe des Vergleichs.

Soll beispielsweise ein Vergleich der Stufe 2 durchgeführt werden, so wird immer auch ein Vergleich der Stufe 1 durchgeführt! Vergleiche niedrigerer Stufe werden also immer auch ausgeführt. Die Angegeben Stufe gibt also die Obergrenze an.

4. Vergleich aller Dateien miteinander. Dabei wird eine eventuelle Transitivität eines Vergleichs nicht ausgenutzt. D.h., wenn A und B gleich sind, und B und C gleich sind, so wird trotzdem auch A mit C verglichen.

   Die Ergebnisse der Vergleiche werden in einer Datenstruktut gespeichert.

5. Sind alle Vergleiche durchgeführt, so werden die Ergebnisse direkt in die miteinander verglichenen Dateien geschrieben.

A.3.2.3.3. Verwendung

vergleiche.pl {-u *`Übungsblatt`*}

A.3.2.3.4. Verwendete Konfigurationsdateien

`conf/xml/uebungsblatt??.xml`

  Konfiguration für das Übungsblatt

A.3.2.3.5. Verwendete Module

`Getopt::Std`

  Auswertung der Kommandozeilen-Parameter

`InfoTool::ParserUebungsblatt`

  Auslesen, in welcher Stufe der Vergleich ausgeführt werden soll. Die Information kommt aus der Datei `conf/xml/uebungsblatt??.xml` (Abschnitt A.1.1.3).

`Konfiguration` (Tool)

  `pfad_webbasis`

    Basispfad für alle anderen Pfade der Web-Schnittstelle

  `datei_konfiguration_uebungsblatt_muster`
  `datei_konfiguration_uebungsblatt_muster_ersetzen`

    Dateiname der Konfigurationsdatei für das Übungsblatt `pfad_toolbasis`

Basispfad für alle anderen Pfade der Administrations-Schnittstelle

`arbeit_pfad`

Arbeits-Pfad, falls nicht anders angegeben

`vergleich_kommentar`

Gibt an, wie der Kommentar mit dem Ergebnis des Vergleichs in die jeweiligen Dateien eingefügt werden soll.

`vergleich_bin`

Kommando, das externes Programm für den Vergleich zweier Dateien ausführt.

A.3.2.4. `bin/autotest.pl`

A.3.2.4.1. Aufgabe

Ruft **testen.pl** (Abschnitt A.2.4.4) mit allen verfügbaren Testsuites für die Lösungen eines Übungsblattes auf.

A.3.2.4.2. Verwendung

autotest.pl {-u *Übungsblatt*} [-d]

Bei Option `-d` werden nur Diskrepanzen protokolliert.

A.3.2.4.3. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`Konfiguration` (Tool)

`pfad_toolbasis`

Basispfad für alle anderen Pfade der Administrations-Schnittstelle `arbeit_pfad`

Arbeits-Pfad, falls nicht anders angegeben

`testsuites_pfad`

Pfad für Testsuites. Hierunter, in dem Verzeichnis für das jeweilige Übungsblatt, müssen die Testdateien liegen.

`bin_pfad`

In diesem Pfad muss das Programm **testen.pl** (Abschnitt A.2.4.4) liegen.

`autotest_kommentar`

Gibt an, wie der Kommentar mit dem Ergebnis des Tests in die jeweiligen Dateien eingefügt werden soll.

A.3.2.5. `bin/zusatzinformationen.pl`

A.3.2.5.1. Aufgabe

Fügt den abgegebenen Aufgaben Zusatzinformationen wie Teammitglieder, Aufgabe, Übungsblatt und maximal erreichbare Punktzahl hinzu. Dies geschieht durch einen Kommentar zu Beginn der Datei. Die Auskommentierung ist dabei vom Dateityp abhängig, der Aufgrund der Dateiendung festgestellt wird.

Trägt, falls noch nicht vorhanden, die Aufgaben des Übungsblatts in die Datenbank ein.

A.3.2.5.2. Verwendung

zusatzinformationen.pl [-b *Arbeits-Pfad*] {-u *Übungsblatt*}

A.3.2.5.3. Verwendete Konfigurationsdateien

`conf/xml/uebungsblatt??.xml`

Konfiguration für das Übungsblatt

A.3.2.5.4. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen und Schreiben folgender Informationen:

- Zuordnung Aufgaben und Übungsblatt, Tabelle aufgabe
- Daten der Studenten, Tabelle student
- Daten der Korrektoren, Tabelle korrektor

`InfoTool::ParserUebungsblatt`

Auslesen von Informationen zu den einzelnen Übungsaufgaben aus der Datei `conf/xml/uebungsblatt??.xml`

`Konfiguration` (Tool)

`pfad_toolbasis`

Basispfad für alle anderen Pfade der Administrations-Schnittstelle

`arbeit_pfad`

Arbeits-Pfad, falls nicht anders angegeben

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`kommentar`

Gibt an, wie der Kommentar mit den Informationen in die jeweiligen Dateien eingefügt werden soll.

A.3.2.6. `bin/aufbereiten.pl`

A.3.2.6.1. Aufgabe

Kopiert die abgegeben Lösungen in die Verzeichnisse der Korrektoren. Abbildung 2-1 zeigt einen Screenshot der dabei angelegten Verzeichnisstruktur.

Zusätzlich wird in den Korrektorenverzeichnis ein kleines bash Skript namens **korrigiert.sh** (Abschnitt A.3.3.1) angelegt, welches beim Ausführen **bewertungen.pl** (Abschnitt A.3.3.2) mit den richtigen Parametern für die Rückgabe des jeweiligen Übungsblatts versorgt.

A.3.2.6.2. Ablauf

1. Auswerten der Kommandozeile
2. Korrektorenliste erstellen
3. Verzeichnisse für die Korrektoren anlegen
4. Lösungen anhand der Kennung und Eintrag in der Datenbank den Korrektoren zuordnen und in ihr jeweilige Verzeichnis korpieren
5. Zugriffsrechte für erstellte Verzeichnisse und Dateien setzen (2770 für Verzeichnisse, 2660 für Dateien)
6. symbolische Links in den Verzeichnissen anlegen (zur Sortierung nach Aufgaben)
7. Shell Skript **korrigiert.sh** anlegen
8. nochmals Zugriffsrechte für erstellte Verzeichnisse und Dateien setzen (2770 für Verzeichnisse, 2660 für Dateien)
9. für die angelegten Dateien das SGID-Bit entfernen (jetzt 0660 für Dateien)

A.3.2.6.3. Verwendung

aufbereiten.pl [-k *`Korrektor-Pfad`*] [-b *`Arbeits-Pfad`*] {-u *`Übungsblatt`*} {-i}

Mit Option `-i` wird beim Überschreiben von Dateien im Korrektor-Verzeichnis nachgefragt.

A.3.2.6.4. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen folgender Informationen:

- Daten der Korrektoren, Tabelle korrektor
- Zuordnung Student und Korrektor, Tabelle korrektor

`InfoTool::ParserUebungsblatt`

Auslesen von Informationen zu den einzelnen Übungsaufgaben aus der Datei `conf/xml/uebungsblatt??.xml`

`Konfiguration` (Tool)

`pfad_toolbasis`

Basispfad für alle anderen Pfade der Administrations-Schnittstelle

`arbeit_pfad`

Arbeits-Pfad, falls nicht anders angegeben

`bin_pfad`

in diesem Verzeichnis muss **bewertungen.pl** (Abschnitt A.3.3.2) liegen

`korrektor_pfad`

Korrektor-Pfad, falls nicht anders angegeben. In dieses Verzeichnis werden die Aufgaben zur Korrektur kopiert

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

A.3.2.7. `bin/zuschicken.pl`

A.3.2.7.1. Aufgabe

Verschickt die Aufgaben falls gewünscht zur Korrektur an den jeweiligen Korrektor.

A.3.2.7.2. Verwendung

zuschicken.pl {-k *`Korrektor-Pfad`*} {-u *`Übungsblatt`*}

A.3.2.7.3. Verwendete Module

`Getopt::Std`

Auswertung der Kommandozeilen-Parameter

`MIME::Lite`

- Versendung der E-Mail mit den zu korrigierenden Aufgaben an den jeweiligen Korrektor

`InfoTool::DBHelper`

Zugriff auf die Datenbank zum Lesen folgender Informationen:

- Daten der Korrektoren, die sich die Aufgaben per E-Mail zuschicken lassen wollen. Tabelle korrektor

`Konfiguration` (Tool)

`pfad_toolbasis`

Basispfad für alle anderen Pfade der Administrations-Schnittstelle

`korrektor_pfad`

Korrektor-Pfad, falls nicht anders angegeben. In dieses Verzeichnis werden die Aufgaben zur Korrektur kopiert

`konfiguration_db_benutzer`
`konfiguration_db_passwort`
`konfiguration_db_name`

Benutzername und Passwort für Zugriff auf die Datenbank `konfiguration_db_name`

`konfiguration_admin_email`
`konfiguration_gruss`
`konfiguration_veranstaltung`

Absender-Adresse, Gruß und Veranstaltung (für Betreff) zum Versenden der E-Mails

A.3.3. Rücknahme der korrigierten Übungsaufgaben

Zur Rücknahme der korrigierten Lösungen gibt es zwei alternative Wege. Die Rücknahme per Web-Formular und die Rücknahme bei bei Korrektur in dem Verzeichnis der Korrektoren. Im folgenden wird auf die Programme für den letzteren Fall eingegangen. Abschnitt A.1.4 beschreibt die Programme für die Rücknahme per Web-Formular.

A.3.3.1. korrigiert.sh

Dieses bash Skript wird von dem Programm **aufbereiten.pl** (Abschnitt A.3.2.6) im Verzeichnis des Übungsblattes im Korrektoren-Verzeichnis erzeugt.

A.3.3.1.1. Aufgabe

Erleichtert die Bedienung von **bewertungen.pl** (Abschnitt A.3.3.2) durch die Korrektoren. Dieses bash Skript ruft **bewertungen.pl** mit den Parametern für das jeweilige Übungsblatt auf.

A.3.3.1.2. Verwendung

korrigiert.sh

Der Aufruf erfolgt ohne Parameter.

A.3.3.2. `bin/bewertungen.pl`

A.3.3.2.1. Aufgabe

siehe Abschnitt A.1.4.2

A.3.3.2.2. Verwendung

bewertungen.pl [-w *`Pfad der korrigierten Lösungen`*] {-u *`Übungsblatt`*}

A.3.3.2.3. Verwendete Module

Bis auf Verwendung von `Konfiguration.pm` (Tool) statt `Konfiguration.pm` (Web) siehe Abschnitt A.1.4.2

A.4. Gemeinsam genutzte Module

Sowohl die Web-Schnittstelle als die Kommandozeilen-Werkzeuge benötigen teilweise die selbe Funktionalität. Aus diesem Grund gibt es einige gemeinsam genutzte Module.

A.4.1. Verarbeitung der XML-Konfigurationsdateien

Die Module zur Verarbeitung der XML-Konfigurationsdateien basieren alle auf `XML::Parser`. Der Parser baut automatische eine Perl-Datenstruktur der XML-Datei auf (aus Hash- und Array-Referenzen). Die Module biete eine bequeme Schnittstelle zu dieser Datenstruktur.

A.4.1.1. `InfoTool/ParserUebungsblaetter.pm`

A.4.1.1.1. Aufgabe

Bereitstellen der Informationen aus `datei_konfiguration_uebungsblaetter`.

A.4.1.1.2. Verwendete Module

`XML::Parser`

Parsen von `datei_konfiguration_uebungsblaetter`

`InfoTool::Datum`

Datumsberechnungen

A.4.1.2. `InfoTool/ParserUebungsblatt.pm`

A.4.1.2.1. Aufgabe

Bereitstellen der Information aus `datei_konfiguration_uebungsblatt??.xml`.

A.4.1.2.2. Verwendete Module

`XML::Parser`

Parsen von `datei_konfiguration_uebungsblaetter`

A.4.2. Datenbankzugriff

Ebenso wie für die XML-Konfigurationsdateien exisitiert auch für die Datenbank ein Modul, um auf die Informationen leichter zugreifen zu können.

A.4.2.1. `InfoTool/DBHelper.pm`

A.4.2.1.1. Aufgabe

Erleichtern des Datenbankzugriffs.

A.4.2.1.2. Verwendete Module

`DBI`

Datenbankunabhängige Schnittstelle für den Datenbankzugriff

A.4.3. Interaktion mit SML

SML stellt nur eine interaktive textbasierte Schnittstelle zur Verfügung. Um über diese Schnittstelle die Lösungen automatisch testen zu können, ist folgendes Modul implementiert worden.

A.4.3.1. `InfoTool/KommSML.pm`

A.4.3.1.1. Aufgabe

Kommunikation mit SML

A.4.3.1.2. Verwendete Module

`IPC::Open2`

bidirektionale Kommunikation mit einem Kind-Prozess (der SML Benutzerschnittstelle)

`IO::Handle`

grundlegende E/A Funktionen

A.4.4. Sonstige Module

Zwei weitere Module, die von verschiedenen Programmen benötigte Funktionalitäten bereitstellen.

A.4.4.1. `InfoTool/CheckPw.pm`

A.4.4.1.1. Aufgabe

Kommuniziert mit dem Programm **pwcheck** der RBG zur Überprüfung von Kennung und Passwort am CIP-Pool.

Kennung und Passwort werden dabei *nicht* in der Kommandozeile von **pwcheck** übergeben, sondern direkt in die Standardeingabe nach Start des Programms geschrieben. Das Ergebnis schreibt **pwcheck** selbst wieder in die Standardausgabe. Es findet also eine vergleichbare Kommunikation mit **pwcheck** wie mit der SML Benutzerschnittstelle statt.

A.4.4.1.2. Verwendete Module

`IPC::Open2`

  bidirektionale Kommunikation mit einem Kind-Prozess (**pwcheck**)

`IO::Handle`

  grundlegende E/A Funktionen

A.4.4.2. `InfoTool/Datum.pm`

A.4.4.2.1. Aufgabe

Konvertierung von Datumsformaten

A.4.4.2.2. Verwendete Module

- **`Time::Local`**. Datums- und Zeitfunktionen

A.4.5. Sonstige Programme

Im folgenden Abschnitt wird auf die Programme unter `info1abg` und auf das Programm der RBG **pwcheck** eingegangen.

A.4.5.1. Programme unter `info1abg`

Die Programme unter `info1abg` dienen Aufruf von SML in einer Art Sandbox. Dazu wird von **ulimit_restricted_bash_SML.sh ulimit_restricted_SML.sh** mit **nice** aufgerufen. Dieses Skript läuft in einer Restricted-Bash und setzt nun ein **ulimit** mit einer Zeitbeschränkung von 900 Sekunden und einer Speicherbeschränkung von 30MB und startet anschließend sml mit **timelimit** auf 30 Sekunden.

Somit ist sichergestellt, dass der SML Prozess zum einen unter dem User info1abg läuft, zum anderen spätestens nach 30 Sekunden oder bei einer Speicherbelegung von 30MB abgebrochen wird.

A.4.5.2. Passwortüberprüfung mit pwcheck

Das Programm **pwcheck** läuft auf dem Rechner `lserv1` der RBG. Nach Start erwartet es die Eingabe einer Kennung und in der nächsten Zeile die Eingabe des dazugehörigen Passworts.

Das Programm liefert folgende Rückgabewerte:

KNWON false

: Kennung ist bekannt, aber das Passwort ist falsch

KNWON true

: Kennung ist bekannt und das Passwort ist richtig

UNKNWON

: Kennung ist unbekannt

Literaturverzeichnis

[Advanced Perl Programming] Sriram Srinivasan, 1997, 1-56592-220-4, O'Reilly & Associates, *Advanced Perl Programming*.

[Apache] Apache HTTP Server Documentation Project, 2000, 1-56592-168-2, *Apache Documentation (http://httpd.apache.org/docs/)*.

[CGI] Shishir Gundavaram, 1996, 1-56592-168-2, O'Reilly & Associates, *CGI Programming on the World Wide Web*.

[MySQL] David Axmark, Michael Midenius, Jeremy Cole, und Paul Dubois, 1997-2000, *MySQL Documentation (http://www.mysql.com/documentation/index.html)*.

[MySQL & mSQL] Randy Jay Yarger, George Reese, und Tim King, 1999, 1-56592-434-7, O'Reilly & Associates, *MySQL & mSQL*.

[Programming Perl] Larry Wall, Tom Christiansen, und Randal L. Schwartz, 1996, 1-56592-149-6, O'Reilly & Associates, *Programming Perl*.