ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik

Institut für Datenbanken und Informationssysteme

in Zusammenarbeit mit dem

Studiendekanat der Medizinischen Fakultät der Universität Ulm

Entwicklung und Realisierung eines Konzepts zur optimierten EDV-Abbildung des Medizin-Curriculums von Studenten der Medizinischen Fakultät der Universität Ulm

Diplomarbeit an der Universität Ulm

Vorgelegt von:

Rainer Mösle

rainer.moesle@uni-ulm.de

13.12.2012

Gutachter:

Prof. Dr. Manfred Reichert

Dr. Ralph Bobrik

Betreuer:

Dipl.-Inf. Rüdiger Pryss

Dipl.-Päd. Claudia Grab

Barbara Eichner

Fassung 13.12.2012

Satz: PDF-LaTeX 2ε

Danksagung

Mein besonderer Dank gilt an dieser Stelle Herrn Prof. Dr. Manfred Reichert vom Institut für Datenbanken und Informationssysteme der Universität Ulm für seine Tätigkeit als Erstgutachter dieser Arbeit. Seine praxisnahen und abwechslungsreich geführten Vorlesungen weckten mein Interesse für weitere Veranstaltungen und Praktika an der Abteilung für Informationssysteme.

Ich bedanke mich ebenfalls herzlichst bei Herrn Dr. Ralph Bobrik für seine Tätigkeit als Zweitgutachter dieser Arbeit.

An dieser Stelle möchte ich einen ganz besonderen Dank an Herrn Dipl.-Inf. Rüdiger Pryss für sein außerordentliches Engagement und die Unterstützung aussprechen, die ich während meines Studiums und dieser Arbeit erfahren habe. Seine Hingabe und Tatkraft als Betreuer verdient meinen allergrößten Dank und Respekt.

Ebenfalls bedanken möchte mich bei Frau Dipl.-Päd. Claudia Grab und Frau Barbara Eichner vom Studiendekanat der Medizinischen Fakultät für die fakultätsübergreifende Unterstützung. Sie standen mir bei der Anforderungsanalyse und während der Umsetzung dieser Arbeit mit fachlichen Ratschlägen und Feedback zur Seite. Nicht zu vergessen Frau Anne Leins, Frau Christina Kollinger und Frau Gabriele Tammer für ihre Anmerkungen und Vorschläge in den Plenumsdiskussionen.

Ein großer Dank gilt Herrn Christian Aschoff vom Kommunikations- und Informationszentrum (kiz) der Universität Ulm für die unkomplizierte und überaus kompetente technische Hilfestellung während dieser Arbeit.

Zuletzt möchte ich bei Herrn Sebastian Jehle bedanken. Er war mir während zahlreichen Vorlesungen nicht nur ein Weggefährte und Ansprechpartner. In ihm habe ich im Laufe meines Studiums auch einen sehr guten Freund gefunden.

Inhaltsverzeichnis

1 Einleitung

Dieses Kapitel gibt einen kurzen Einblick in die Thematik der vorliegenden Diplomarbeit. Zunächst werden die Beweggründe für die Diplomarbeit angesprochen, darauf folgend wird im Abschnitt 1.2 ein Überblick über den Aufbau der Arbeit gegeben. Sie entstand am Institut für Datenbanken und Informationssysteme der Universität Ulm, nachfolgend DBIS genannt, in Zusammenarbeit mit dem Studiendekanat der Medizinischen Fakultät.

1.1 Motivation und Zielsetzung

Die Notenverwaltung ist ein zentrales Thema im Arbeitsablauf des Studiendekanats. Mit einem speziell auf die Bedürfnisse der Medizinischen Fakultät angepassten und etablierten Notenverwaltungssystem, nachfolgend Blaues System genannt, werden Noten erfasst und Studenten verwaltet. Von Studenten entwickelt, befindet sich das System auf dem Entwicklungsstand des Oktobers 2009, wie aus der Dokumentation der Änderungen im Blauen System entnommen werden kann [1]. Das System bietet den Mitarbeitern des Dekanats eine einfache und schlichte Oberfläche zur Verwaltung von Studenten. Es können nicht nur Noten oder Wahlfächer erfasst werden, sondern auch zahlreiche Nachweise automatisch erstellt werden. Durch wechselnde Prüfungsordnungen, neue Fachbereiche und Anforderungen, die erst im Laufe der Zeit im produktiven Einsatz deutlich wurden, muss sich das System nun weiterentwickeln. Durch den langjährigen Einsatz gibt es nicht nur seitens der Prüfungsverwaltung, sondern auch von den Mitarbeitern des Dekanats, konkrete Rückmeldungen, wie die Arbeit mit dem System vereinfacht und verbessert werden kann. Da die Notenverwaltung speziell auf die Medizinische Fakultät zugeschnitten ist und seitens der Universität eine einheitliche Lösung für alle Fakultäten gewünscht und kommuniziert wird, sieht sich das Kommunikations- und Informationszentrum der Universität Ulm, nachfolgend kiz[1] genannt, nicht in der Pflicht, das Blaue System weiterzuentwickeln.
Ziel dieser Arbeit ist es daher, in enger Zusammenarbeit mit dem Studiendekanat,

[1] Zuständig für die Kommunikations- und Informationsstruktur der Universität Ulm. Zu den Aufgabenbereichen gehören unter anderem: Bibliothek, Informationstechnik und Medien.

die neuen Anforderungen zu erfassen und das System weiterzuentwickeln. Der Schwerpunkt liegt dabei auf der optimierten, systemübergreifenden Abbildung der Studienordnung. Durch die Zusammenarbeit mehrerer Systeme soll zum einen die automatische Konsistenzsicherung optimiert werden, zum anderen ein Mehrwert für Studenten geschaffen werden.

1.2 Aufbau der Arbeit

Diese Arbeit ist in neun Kapitel aufgeteilt. Eine Übersichtsgrafik findet sich in Abbildung 1.1. In Kapitel 2 werden konkurrierende Studierendenverwaltungssysteme vorgestellt, die eine Prüfungsverwaltung anbieten. Anschließend wird in Kapitel 3 das Medizinstudium näher betrachtet. Das hieraus gewonnene Verständnis für das Einsatzumfeld des Blauen Systems hilft bei der näheren Betrachtung des Systems in Kapitel 4. Hier werden die verwendeten Technologien und die Funktionsweise anhand verschiedener Aufgabengebiete vorgestellt. Im darauffolgenden Kapitel 5 werden die Anforderungen an die Erweiterung des Blauen Systems und dementsprechend die Aufgabenstellungen dieser Arbeit dargestellt und ausführlich beschrieben. Kapitel 6 gibt zunächst eine Übersicht über die verwendeten Systeme und stellt diese in einen gemeinsamen Zusammenhang, bevor in Kapitel 7 konkrete Implementierungsdetails dieser Arbeit veranschaulicht werden. Abschließend wird in Kapitel 8 abgeglichen, inwieweit die definierten Anforderungen im Rahmen dieser Arbeit erfüllt wurden. Kapitel 9 gibt schließlich eine Zusammenfassung dieser Arbeit und einen Ausblick auf künftige Erweiterungen und noch nicht umgesetzte Funktionen.

1 Einleitung	1.1 Motivation und Zielsetzung	1.2 Aufbau der Arbeit			
2 Related Work	2.1 Hochschul-Informations-System	2.2 CampusNet	2.3 CAMPUSOnline	2.4 FlexNow	2.5 PRIMUSS
	2.6 FHComplete	2.7 Zusammen-fassung	2.8 OpenSource als Alternative?	2.9 Fazit	
3 Medizinstudium	3.1 Das Studium	3.2 Fazit			
4 Das Blaue System	4.1 Technologien	4.2 Funktionen	4.3 Fazit		
5 Anforderungen	5.1 Zusammen-arbeit: DBIS, Medizin & kiz	5.2 Vorlage des Studiendekanats	5.3 Blaues System	5.4 Corona	5.5 MedicUlm
	5.6 WebService	5.7 Moodle	5.8 Schwierigkeiten	5.9 Fazit	
6 Gesamt-architektur	6.1 Datenbank Blaues System	6.2 Blaues System und Corona	6.3 MedicUlm, WebServices und Blaues System	6.4 Fazit	
7 Implementierung	7.1 Blaues System	7.2 Corona	7.3 MedicUlm	7.4 getGrades WebService	7.5 Problem-stellen im Blauen System
	7.6 Verbesse-rungsbedarf	7.7 Fazit			
8 Abgleich der Anforderungen	8.1 Blaues System	8.2 Corona	8.3 MedicUlm	8.4 WebService	8.5 Moodle
	8.6 Fazit				
9 Zusammen-fassung und Ausblick	9.1 Zusammen-fassung	9.2 Ausblick			

Abbildung 1.1: Aufbau der Arbeit

2 Related Work

Im Bereich der Studierendenverwaltung finden sich, abgesehen von universitären Eigenentwicklungen wie dem Blauen System, auch Verwaltungssysteme professioneller Anbieter. Diese sind meist in sogenannte Hochschulinformationssysteme, auch Campus-Management-Systeme genannte IT-Systeme, eingegliedert [2]. Diese sind modular aufgebaut. Der Funktionsumfang unterscheidet sich zwar je nach Anbieter, allerdings haben die verschiedenen Lösungen meist eine Schnittmenge an Modulen gemein. Diese umfassen nicht nur studiumsbegleitende Funktionen wie die Studierendenverwaltung, studienorganisatorische Verwaltungsfunktionen wie dem Lehrraummanagement, Steuerung von Zulassungsverfahren oder Veranstaltungsplanungen, sondern auch die Verwaltung von Prüfungsanmeldungen [2]. In den folgenden Abschnitten werden verschiedene Systeme vorgestellt, die dem Blauen System im Funktionsumfang ähneln. Aus diesen Systemen werden die für diese Arbeit relevanten Module und Funktionen vorgestellt.

2.1 HIS - Hochschul-Informations-System

Das HIS, dem seit dem Jahre 1975 Bund und Länder als Gesellschafter vorstehen, ist ein nicht gewinnorientiertes Unternehmen. Es bietet ein Leistungsangebot für deutsche Hochschulen, welches unter anderem einen Schwerpunkt auf die Hochschulverwaltung legt. Hierunter fällt auch die Prüfungs-, beziehungsweise Studierendenverwaltung [3]. Das Leistungsangebot umfasst eine Vielzahl an Modulen. Nachfolgend findet sich eine Auswahl:

- QIS - Qualitätssteigerung der Hochschulverwaltung im Internet durch Selbstbedienung
- LSF - Lehre, Studium und Forschung
- SOS - Studierenden-Management
- POS - Prüfungsverwaltung
- ZUL - Zulassung und Studium
- SVA - Personal- und Stellenmanagement

Die an der Universität Ulm eingesetzten Module sind unter anderem das Studierenden-Management (SOS), die Prüfungsverwaltung (POS), deren Webequivalent (QIS POS) und die Webanwendung für Lehre, Studium und Forschung (QIS LSF). Der Zusatz QIS bezeichnet Module, die im Internet Funktionen für Studierende und Mitarbeiter bereitstellen. Dadurch minimiert sich nicht nur der Serviceaufwand für die Mitarbeiter, sondern macht Studierende unabhängig von Bürozeiten [4]. Die Abbildung 2.1 zeigt die Leistungsansicht eines Studenten des QISSOS Moduls. Die angesprochenen Kernmodule werden nun näher beschrieben.

HTML-Ansicht Ihrer erbrachten Leistungen

Wenn Sie auf den info-Button neben der erzielten Note klicken, erhalten Sie einen „Klassenspiegel"

Offizielle Bescheinigungen über Studien- und Prüfungsleistungen erhalten Sie unter "Prüfungsverwaltung"

Stammdaten des Studierenden

Name des Studierenden	Max Mustermann
Geburtsdatum und -ort	22.08.1985 in Musterstadt
(angestrebter) Abschluss	Diplom
Fach	Medieninformatik
Matrikelnummer	123456
Anschrift	Musterstrasse 42, 12345 Musterstadt

Prüfungsnr.	Prüfungstext	Semester	Note	Status	LP	Vermerk	Versuch	Prüfungsdatum
140	Mathematik für Informatiker	WiSe 06/07	1,7 info	bestanden	24		1	30.03.2007
151	Medienpädagogik	SoSe 07	1,3 info	bestanden	6		1	19.07.2007
110	Praktische Informatik	WiSe 07/08	5,0 info	nicht bestanden	0		1	08.10.2007
130	Mediale Informatik - Interaktive Systeme	WiSe 07/08	1,3 info	bestanden	14		1	28.02.2008
141	Allgemeine Betriebswirtschaft I	WiSe 07/08	2,3 info	bestanden	0		1	25.02.2008
150	Anwendungsfach	WiSe 07/08	1,8 info	bestanden	6		1	25.02.2008

Abbildung 2.1: Die Leistungsansicht eines Studenten

2.1.1 Studierenden-Management (SOS)

Das SOS Modul verwaltet den Verlauf eines Studenten von der Einschreibung bis hin zur Exmatrikulation. Es bietet einen umfassenden und schnellen Einblick in den Studienfortschritt. Zudem stehen Funktionen zur Erstellung und dem Export von Listen und Statistiken oder der Verwaltung von Chipkarten zur Verfügung. Die Benutzeroberfläche kann an die jeweiligen Bedürfnisse angepasst und erweitert werden [5].

2.1.2 Prüfungsverwaltung (POS)

Mit dem SOS verdrahtet ist das POS Modul, das sämtliche mit den Prüfungen relevanten Verwaltungsabläufe instrumentalisiert. Ein wichtiger Bestandteil neben der Erstellung von Prüfungs-, Raum- sowie Personalplänen, und der Erfassung von Noten, ist die Abbildung von Prüfungsordnungen. Dazu gehört auch das neue

Credit-Point System für Bachelor- und Masterstudiengänge [6]. Zudem bietet die Prüfungsanmeldung die Möglichkeit, Zulassungvoraussetzungen und Fristen zu kontrollieren [5].

2.1.3 Lehre, Studium und Forschung (LSF)

Das LSF ist eine Informationsplattform, unter anderem für Lehrveranstaltungen und Studiengänge. Studenten haben die Möglichkeit, sich ihren Stundenplan zusammenzustellen. Die Daten können von den verschiedenen Fachbereichen selbst gepflegt werden. Dazu zählen neben den Vorlesungsverzeichnissen auch die für Veranstaltungen relevanten Informationen, wie Uhrzeiten, Raumbelegungen und Dozenten [5].

2.1.4 Verbreitung

Die HIS-Software wird an über 220 deutschen Hochschulen in Anspruch genommen [7]. Neben der Fachhochschule und der Universität Ulm ist sie auch an Deutschlands größten Universitäten im Einsatz. Dazu zählt beispielsweise die Fernuniversität Hagen, die LMU München oder auch die Goethe-Universität Frankfurt am Main [8].

2.2 CampusNet

Im Jahre 1999 entschieden sich die ersten Hochschulen für Produkte der Datenlotsen Informationssysteme GmbH, deren Kernprodukt das CampusNet System ist [9]. Als Campus-Management-System hat CampusNet den Student Lifecycle, sprich die Abbildung des studentischen Lebenszyklus, als Leitmotiv.

2.2.1 Student Lifecycle

Der Student wird von der eigentlichen Interessensfindung, über den gesamten Studienverlauf, bis hin zum Alumni begleitet. Es stehen die benötigten Funktionen zur Verfügung, die eine optimale Verwaltung des Studenten ermöglichen. Hierbei sei neben der eigentlichen Zulassung, einem Veranstaltungs- und Lehrendenmanagement, Raumplanungsmöglichkeiten, auch die eigentlichen Studierendenverwaltung erwähnt. Diese wird ergänzt durch ein Prüfungsmanagement, das Leistungsnachweise, Teilnehmerlisten als auch die reine Notenerfassung ermöglicht. Prü-

fungsergebnisse können vom Prüfer selbst oder von Vertretern eingetragen werden. Diese müssen dann allerdings vom Prüfer freigegeben werden. Der Student selbst kann über ein Webportal auf verschiedene Services zugreifen. Zum einen kann er die angebotenen Module und Veranstaltungen einsehen und sich über ein TAN-Verfahren anmelden. Zum anderen können Prüfungsleistungen in einer Leistungsübersicht jederzeit eingesehen werden [9].

2.2.2 Verbreitung

CampusNet kommt an über 60 Hochschulen in 12 Bundesländern zum Einsatz. Das Campus-Management-System kommt unter anderem in der Johannes Gutenberg-Universität Mainz, der Technischen Universität Dresden und der Technischen Universität Berlin zum Einsatz [10].

2.3 CAMPUSOnline

Mit dem Ziel einer gemeinsamen Datenbasis, basierend auf einem zentralen Datenmodell, wurde an der Technischen Universität Graz im Jahre 1997 erstmals mit der Entwicklung eines Informationsmanagement-Systems begonnen. 1998 wurde schließlich TUGonline, ein Vorläufer des heutigen CAMPUSOnline, publiziert. 2004 folgte CAMPUSOnline, das zunächst an österreichischen Hochschulen und Universitäten zum Einsatz kam [11]. Das System ist unterteilt in 6 Module [12]:

- Student Lifecycle
- Basis Ressourcen Management
- Customer Support Services
- Forschung
- Integration und Schnittstellen
- Administration und CRM

Die für diese Arbeit zu berücksichtigenden Module werden nachfolgend beschrieben.

2.3.1 Student Lifecycle

Dieses Modul entspricht - ähnlich dem HIS SOS-Modul in Abschnitt 2.1.1 - einer Studierendenverwaltung. Sie begleitet die Studierenden vom Auswahlverfahren an

über die gesamte Dauer des Studiums. Es enthält eine Prüfungsverwaltung, die neben Prüfungsanmeldungen auch Noten, Anerkennungen und Abschlussarbeiten erfasst. Neben einem Lehrveranstaltungsangebot, über das sich auch Studienordnungen abbilden lassen, finden sich Evaluations- und Alumnifunktionen [12].

2.3.2 Basisressourcen-Management

Dieser Bereich ist die Kernkomponente des Systems. Er erlaubt eine umfassende Verwaltung von Mitarbeitern, externem Personal und Organisationen. Basierend auf einem Organisations-Metamodell können sowohl Organisationen als auch Benutzerrechte zentral gesteuert werden. Anhand verschiedener Profile, wie Studierende oder Mitarbeiter, wird eine Administration von Benutzerkonten aller verwendeten Module ermöglicht [12].

2.3.3 Verbreitung

Heute ist CAMPUSOnline das marktführende Campus-Management-System Österreichs. Unter den 35 Hochschulen, die das System verwenden, finden sich auch Hochschulen in Deutschland. Neben dem Produktiveinsatz an der Technischen Universität München befindet es sich an der Universität zu Köln und der Universität Stuttgart bereits in der Einführungsphase [13].

2.4 FlexNow

FlexNow ist im Gegensatz zu den bisher vorgestellten Verwaltungssystemen rein auf die eigentliche Prüfungsverwaltung spezialisiert. Es wird seit 1994 vom Lehrstuhl für Wirtschaftsinformatik der Universität Bamberg entwickelt und erhielt 1996 den Förderpreis des deutschen Stifterverbands für die deutsche Wissenschaft [14]. Das System besteht aus vier Modulen:

- Prüfungsordnungsmodul
- Prüfungsamtmodul
- Prüfungsdurchführungsmodul
- Lehrstuhlmodul

Das System begleitet den Prüfungsverlauf von der Erstellung der Prüfungen, der darauffolgenden internetbasierten Anmeldung für Studenten, über die Raum- und

Zeitplanung, bis hin zur eigentlichen Eintragung von Prüfungsergebnissen. Außerdem können auch Nachweise und Zeugnisse ausgestellt werden [15]. Da der Fokus dieser Arbeit auf den Funktionalitäten des gesamten Systems liegt, wird nun kein einzelnes Modul herausgestellt, sondern das System im Ganzen betrachtet.

2.4.1 Aufbau & Funktionen

FlexNow basiert auf einem Client-Server System. Die Clients greifen je nach Rechtevergabe schreibend oder lesend, beziehungsweise direkt oder indirekt auf die Daten des Servers zu. Hier wird zwischen internen Mitarbeitern des Prüfungsausschusses, des Lehrstuhls oder des Prüfungsamtes, aber auch Studenten unterschieden. Studenten haben nur über einen WWW-Server Zugriff auf die Datenbank, Mitarbeiter können direkt mit dem Datenbank-Server arbeiten [16]. Nachdem im Prüfungsordnungsmodul über Metadaten entsprechende Prüfungsordnungen formalisiert wurden, kann im Prüfungsdurchführungsmodul die eigentliche Prüfungsverwaltung durchgeführt werden. Diese besteht unter anderem aus der Erstellung von Prüfungen oder der Raum- und Zeitplanung. Hier können für Prüfungen, für die im Lehrstuhlmodul Noten eingetragen wurden, Bescheinigungen und Zeugnisse erstellt werden. Im Lehramtmodul hingegen können die Stammdaten und Prüfungsdaten der Studenten verwaltet werden [17]. Diese Stammdaten lassen sich auch aus dem bereits in Abschnitt 2.1.1 vorgestellten HIS Modul HIS SOS extrahieren [18].

2.4.2 Verbreitung

Das FlexNow System kommt an mittlerweile 15 Hochschulen, darunter 13 Universitäten, zum Einsatz. Zu den größten Universitäten zählen hierbei die Universität Hamburg, die Westfälische Wilhelms-Universität Münster und die Justus-Liebig-Universität Gießen [19].

2.5 PRIMUSS

Das *Prüfungs-, Immatrikulations- und Studentenverwaltung System*, kurz PRIMUSS, wird seit 2001 von sechs deutschen Hochschulen entwickelt [20]. Diese haben sich in einem vertraglich geregelten Verbund zusammengeschlossen. Durch die enge Zusammenarbeit kann in die Weiterentwicklung des Systems ein umfänglicher Erfahrungsschatz einfließen. Hiervon profitiert der Verbund nicht nur von

den Rückmeldungen und Evaluationen aller angebundenen Hochschulen, auch das Entwicklerteam selbst kann auf Wissen, Kenntnisse und Ressourcen einzelner Hochschulen zurückgreifen [21]. Durch die gemeinsame Arbeit ist dementsprechend auch die Serverstruktur so gestaltet, dass eine enge Verdrahtung zwischen den Hochschulen vorhanden ist. Der Hauptserver steht in der Hochschule Coburg, der die Daten sowohl lokal als auch an die *Slave Server* der beteiligten Hochschulen spiegelt. Sachbearbeiter haben auf ihren lokalen *Slave Servern* einen lesenden Zugriff. Schreibzugriffe erfolgen immer auf dem *Master Server*, der diese Daten wiederum in allen Hochschulen spiegelt. Die Online-Dienste für PRIMUSS Kunden werden über einen Web-Server an der Hochschule Coburg bereitgestellt [22].

Der PRIMUSS-Verbund ist in 6 Fachbereiche unterteilt [23]:

- PRIMUSS - SD: Server und Datenbanken
- PRIMUSS - KS: Kernsystem
- PRIMUSS - RS: Raum- und Stundenplanung
- PRIMUSS - LP: Lehrveranstaltung- und Prüfungsplanung
- PRIMUSS - Online: Online-Dienste
- PRIMUSS - Mobile: Applikation für Android und Apple iOS

Der Bezug zum Blauen System lässt sich über den Student Lifecycle herstellen.

2.5.1 Student Lifecycle

Wie in den bereits vorgestellten Projekten liegt der Fokus auch bei PRIMUSS auf dem studentischen Lebenszyklus. Dieser beginnt bereits vor dem eigentlichen Studium, indem für Schüler und potentielle Bewerber gezielt Informationen bereitgestellt werden und anschließend bei der Bewerbung unterstützt werden. In der eigentlichen Studienphase stellt das System Funktionen zur Studenten- und Prüfungsverwaltung bereit. Hier können Studenten online über das PRIMUSS-Portal unter anderem ihren Status überprüfen, Bescheide drucken oder sich durch die Anbindung an das Kassenbuch der Hochschule online zum Semester rückmelden. In der Prüfungsverwaltung stehen neben der klassischen Prüfungsanmeldung und Noteneinsicht auch eine Einschreibungsfunktion zur Verfügung, die eine Modulverlosung für Studiengänge mit einer hohen Zahl an Studenten erlaubt [24].

2.5.2 Verbreitung

Zu den Verbundpartnern zählen insgesamt 6 Hochschulen der Standorte Coburg, Amberg-Weiden, Freiburg, Hof, Ingolstadt und Nürnberg. Weitere vier Hochschulen, darunter die Fachhochschule München und Regensburg, zählen zu ihren Kunden [21].

2.6 FH Complete

Das 2004 veröffentlichte, freie Softwarepaket *FH Complete* wird von der Fachhochschule Technikum Wien entwickelt [25]. Es ist unter GPL[1] lizenziert und darf somit kostenfrei vervielfältigt und angepasst werden [26]. Auch hier steht der Student im Mittelpunkt. Er soll bereits vor dem Studium und auch während des Studienzykluses an der Hochschule begleitet werden. Das Paket besteht aus 5 Modulen, die an eine zentrale Datenbank gebunden sind [27]:

- CIS - Campus Informations-System
- FAS-ONLINE - Freies Administrations-System
- TEMPUS - System zur Stundenplanung
- WAWI - Bestellwesen & Warenwirtschaft
- VILESCI - Systemverwaltung & -wartung

Nachfolgend wird das FAS-Online Modul näher beleuchtet, da es, wie die bereits vorgestellten Systeme, den Student Lifecycle als Grundlage hat.

2.6.1 FAS-ONLINE - Freies Administrations-System

Dieses Modul bietet eine Verwaltung aller studienrelevanten Daten während des gesamten Studiums. Für Studenten werden Noten, Prüfungen oder Projektarbeiten erfasst. Plausiblitätsprüfungen erleichtern Mitarbeitern den Verwaltungsaufwand. Zudem gibt es unter anderem Funktionen zur Erstellung von Lehrveranstaltungen, beziehungsweise einer Kostenübersicht.

2.6.2 Verbreitung

Neben der Fachhochschule Technikum Wien, verwendet lediglich die Fachhochschule St. Pölten die FH Complete Software.

[1] Software-Lizenz, die festlegt, inwiefern diese Software genutzt, verbreitet und geändert werden darf.

System	OpenSource	Partner
HIS	Nein	220
CampusNet	Nein	60
CAMPUSOnline	Nein	35
FlexNow	Nein	15
PRIMUSS	Nein	6
FH Complete	Ja	2

Tabelle 2.1: Überblick über die vorgestellten Systeme

2.7 Zusammenfassung

In den vorhergehenden Abschnitten dieses Kapitels wurden verschiedene Systeme vorgestellt, die eine reine Prüfungsverwaltung anbieten oder als Modul im Gesamtsystem eingebettet sind. Tabelle 2.1 zeigt zusammenfassend die Verbreitung der Systeme an deutschsprachigen Hochschulen und ob sie als OpenSource Software verfügbar sind. Auffallend ist, dass alle Systeme bis auf FlexNow, ein ganzheitliches System bereitstellen. Stand-Alone Systeme, die auf einen Verwaltungsaspekt spezialisiert sind, können zwar einfach in ein bestehendes Verwaltungssystem eingebettet werden. Allerdings muss ein höherer Aufwand betrieben werden, wenn Schnittstellen zu bestehenden Systemen gefordert sind. Hier lässt sich auch ein Bezug zum Blauen System herstellen. Vormals einzig zum Zwecke der Notenverwaltung entwickelt, wurde im Laufe des Produktiveinsatzes der Bedarf an Schnittstellen zu bestehenden Systemen wie Corona oder Moodle geweckt. In Kapitel 5 wird näher auf diese Anforderungen und Systeme eingegangen.

2.8 OpenSource als Alternative?

Hochschulinformationssysteme, beziehungsweise Campus-Management-Systeme, stellen für Hochschulen einen hohen Kostenfaktor dar. Daher stellt sich die Frage, ob es adäquate OpenSource Software gibt, die die meisten Anforderungen erfüllt. Dass OpenSource Software an Hochschulen in anderen Bereichen durchaus Erfolg haben kann, zeigt das Beispiel der freien Lernplattformen Moodle [28] oder ILIAS [29]. Hier handelt es sich allerdings um Module, die nicht eng verzahnt mit anderen Modulen sind. Neben der in Abschnitt 2.6 vorgestellten FH Complete Software, wäre noch die englischsprachige FreeSMS Software (Free Student Management System) [30] zu erwähnen.

Eine der wichtigsten Anforderungen an ein Campus-Management-System ist die Anpassbarkeit an die eigenen Bedürfnisse und stetige Verbesserungen. Der Code muss nicht nur der aktuellen Entwicklung standhalten, es müssen auch aufkommende Sicherheitslücken schnell geschlossen werden können. Hierfür erfordern externe OpenSource Lösungen einen hohen Initialaufwand und den steten Support der Community. Auch muss seitens der Hochschule Personal für Anpassungen und Weiterentwicklungen bereitgestellt werden. Daher schlagen Hochschulen, wie das Technikum Wien mit FH Complete, den Weg ein, ein solches Projekt selbst zu stemmen, um schnell auf neue Anforderungen reagieren zu können. Somit entstehen außer Personalkosten keine weiteren Kosten für Lizenzen, Support oder ähnliches. Das Beispiel der FH Complete Software zeigt allerdings auch, dass der OpenSource Gedanke schnell in den Hintergrund rücken kann. Die Projektseite, unter der auch unter anderem eine Demo des FAS-ONLINE Moduls angeboten wird, wird offensichtlich nicht mehr gepflegt. Das FAS Modul weist einen leeren Datenbestand auf, das Tempus-Modul weist mehrere sichtbare Syntaxfehler auf [31]. Eine Projektversionierung mit Git oder eine sonstige Downloadmöglichkeit des Source Codes wird nicht angeboten. Dies ist möglicherweise auf eine mangelnde Rückmeldung anderer Hochschulen zurückzuführen. Diese Tatsache erweckt den Anschein, dass eine für Hochschulen lukrativere Variante der kommerzielle Vertrieb von Software ist. Ein Beispiel hierfür wäre das CampusOnline Projekt der TU Graz, das in Abschnitt 2.3 beschrieben wurde.

2.9 Fazit

Es wird beim Vergleich zwischen den vorgestellten Systemen und dem Blauen System deutlich, dass das Blaue System in puncto Funktionsumfang nicht mithalten kann. Allerdings sollte man dabei nicht aus den Augen verlieren, dass die Entwicklung des Systems speziell für die Anforderungen des Studiendekanats erfolgte. Eine Komplettlösung, beispielsweise vom Umfang eines HIS oder CampusOnline Systems, war nicht erwünscht, beziehungsweise nicht erforderlich. Auch wären für ein solches Projekt mehr Ressourcen notwendig. Dafür ist das Blaue System genau auf die Bedürfnisse der Mitarbeiter des Dekanats zugeschnitten. Bevor das System in Kapitel 4 näher vorgestellt wird, bietet das folgende Kapitel einen Einblick in das Medizinstudium. Dadurch kann ein grundlegendes Verständnis für die folgenden Kapitel gewonnen werden.

3 Medizinstudium

Der Leistungsumfang des Blauen Systems, als auch die neuen Anforderungen an das System, sind eng an das Medizinstudium gekoppelt. Daher wird in diesem Abschnitt das Studium der Medizin in seinen Grundzügen erläutert.

3.1 Das Studium

An der Medizinischen Fakultät der Universität Ulm werden drei verschiedene Studiengänge der Medizin angeboten. Diese Studiengänge der Human-, Zahn- und Molekularmedizin werden nun näher erläutert, um ein Grundverständnis für das Blaue System zu schaffen, und die neuen Anforderungen einordnen zu können. Abbildung 3.1 zeigt eine Übersicht der Medizin-Studiengänge der Universität Ulm.

3.1.1 Humanmedizin

Das Studium der Humanmedizin mit einer Regelstudienzeit von 12 Semestern gliedert sich in drei Teile. Dem vorklinischen und klinischen Studienabschnitt und einem praktischen Jahr. Anschließend folgt die Approbation. Der vorklinische Abschnitt, der zwischen dem 1. bis 4. Semester angesiedelt ist, schafft wichtige naturwissenschaftliche, medizinische und praxisorientierte Grundlagen für den weiteren Studienverlauf. Hier können sich die Studenten einen Überblick über die verschiedenen Anwendungsfelder der Medizin verschaffen, um sich im weiteren Verlauf des Studiums in ein Fachgebiet zu vertiefen. Hierzu gehören unter anderem Seminare mit klinischen Bezügen oder integrierte Seminare, die einen Schwerpunkt auf die Praxisarbeit legen. Ab dem 4. Semester werden sogenannte Querschnittsfächer angeboten. Diese sollen Zusammenhänge zwischen Veranstaltungen herstellen und fächerübergreifendes Denken fördern. Im vorklinischen und klinischen Studienabschnitt kann aus einem Veranstaltungsangebot jeweils ein Wahlfach abgelegt werden. Nach dem 4. und 12. Semester findet jeweils eine staatliche ärztliche Prüfung statt [32].

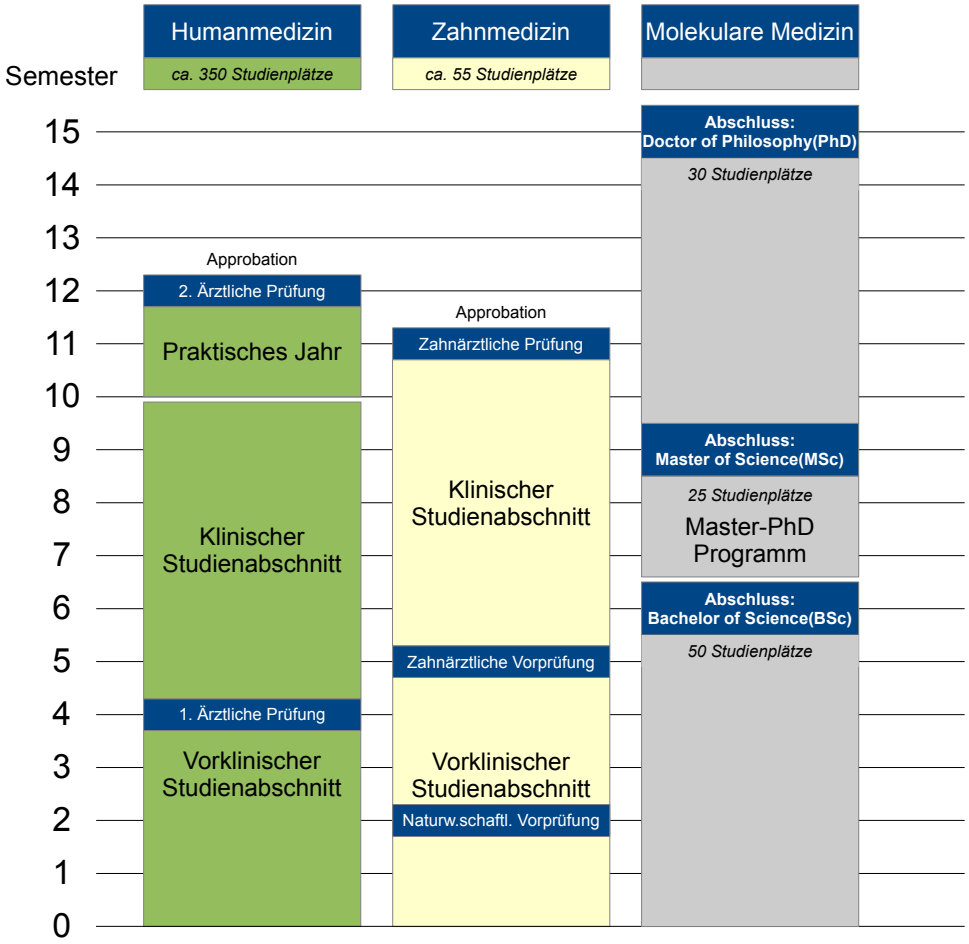


Abbildung 3.1: Übersicht über die Medizin-Studiengänge

3.1.2 Zahnmedizin

Im Gegensatz zur Humanmedizin ist ein Studium der Zahnmedizin bereits nach einer Regelstudienzeit von 10 Semestern beendet. Auch hier unterteilt sich der Studienverlauf in den vorklinischen und klinischen Studienabschnitt. Das Praktische Jahr entfällt. Neben der zahnärztlichen Vorprüfung am Ende des 5. Semesters und der zahnärztlichen Prüfung am Ende des Studiums, findet in der Vorklinik zusätzlich am Ende des 2. Semesters die naturwissenschaftliche Vorprüfung statt. Nach Abschluss des Studiums kann wiederum die Approbation beantragt werden [32].

3.1.3 Molekularmedizin

Das Studium der Molekularmedizin ist in drei Studienabschnitte aufgeteilt. Der Bachelorabschnitt wird in sechs Semestern abgelegt und verbindet die Gebiete der Biologie und der Medizin. Hier werden Grundlagen für Tätigkeiten im Bereich der Forschung und Entwicklung in der molekularen Medizin vermittelt [33]. Aufbauend auf dem Bachelorstudium kann, sofern die Zulassungskriterien erfüllt werden, ein Masterstudium begonnen werden. Dieses dauert drei Semester und ist englischsprachig. Für besonders qualifizierte Studenten wird zusätzlich der Promotionsstudiengang *International PhD Programme in Molecular Medicine* angeboten, der sich über sechs Semester erstreckt [34]. Alle Studienabschnitte haben eine Abschlussprüfung gemein.

3.1.4 Studienordnung

Jedem Studienabschnitt ist eine Studienordnung zugeordnet, die bestimmte Zulassungskriterien definiert. Dazu gehören unter anderem Anmeldefristen oder Voraussetzungen, um an Veranstaltungen teilnehmen zu dürfen. Die Studienordnung der Humanmedizin des vorklinischen Studienabschnitts begrenzt beispielsweise in §4, Abschnitt 2, die Anzahl der Prüfungsversuche auf 5 Versuche, den Zeitraum von Wiederholungsprüfungen auf maximal 24 Monate. Weiter sind im §2, Abschnitt 2, Voraussetzungen definiert, die die Zulassung für die Teilnahme an Lehrveranstaltungen regeln. Beispielsweise setzt die Teilnahme an der Veranstaltung des Kursus der Makroskopischen Anatomie den Nachweis der drei folgenden Scheine voraus [35].

- Seminar Anatomie
- Praktikum der medizinischen Terminologie
- Nachweis der Untersuchung nach der Biostoffverordnung

3.2 Fazit

Das Medizinstudium ist in drei verschiedene Studiengänge aufgeteilt, die sich nicht nur durch unterschiedliche Abschlüsse unterscheiden, sondern auch durch verschiedene Studienordnungen. Zusätzliche Kriterien, wie die Voraussetzungen für die Teilnahme an Prüfungen, steigern die Komplexität bei der Abbildung des Studiums. Der Einstieg fällt dadurch für Entwickler ohne Hintergrundwissen schwer, zumal die einzelnen Funktionen des Systems nicht weiter dokumentiert sind. Daher

werden im folgenden Kapitel die verwendeten Technologien, als auch die Funktionalitäten des Blauen Systems anhand verschiedener Themengebiete näher erklärt.

4 Das Blaue System

Das Blaue System ist seit 2006 im Studiendekanat der Medizinischen Fakultät der Universität Ulm im Einsatz. Der Schwerpunkt des Systems liegt in der reinen Notenverwaltung der Medizinstudenten während ihres Studiums an der Universität Ulm. Dabei bietet es den Mitarbeitern eine einfache und verständliche Schnittstelle zur Erfassung und Speicherung von Daten und Prüfungsleistungen.

4.1 Technologien

Das Blaue System ist vor unbefugtem Zugriff nicht nur mit einem Login-Verfahren, wie in Abbildung 4.1 zu sehen, geschützt, sondern auch der Zugriff auf den Localhost und das Subnetz des Studiendekanats beschränkt. Nach erfolgreichem Login steht dem Benutzer die volle Funktionalität des Systems zur Verfügung. In diesem Abschnitt werden die Technologien vorgestellt, die im Blauen System verwendet werden.

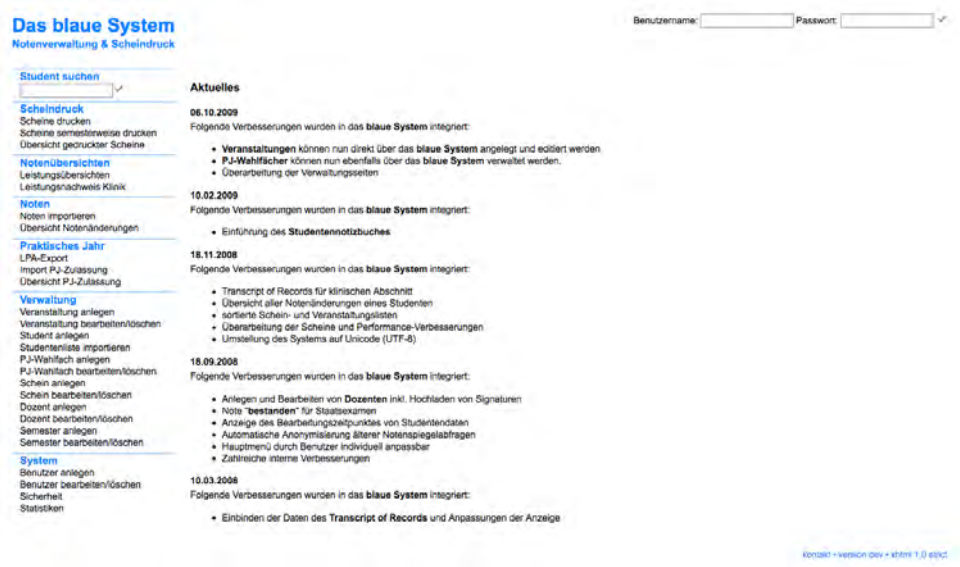


Abbildung 4.1: Login-Bereich des Blauen Systems

4.1.1 PHP

Vorausgesetzt wird eine PHP-Installation ab Version 4.3.10. Das System basiert größtenteils auf nicht-objektorientierten PHP-Dateien. Beim Aufruf eines Menüpunktes wird die dafür hinterlegte PHP Datei an den PHP Interpreter übergeben und verarbeitet. Dabei ist die Programmlogik von der eigentlichen Präsentation getrennt.

4.1.2 MySQL

Voraussetzung ist MySQL ab Version 4.1.11. Die benötigten Daten für die Ausgabe werden aus der MySQL Datenbank selektiert, je nach Funktion mit weiteren Daten angereichert und an die Smarty Template Engine übergeben. Diese Smarty Bibliothek wird im nächsten Abschnitt näher beschrieben.

4.1.3 Smarty Template Engine

Diese freie PHP-Bibliothek ist unter LGPL lizenziert. Sie kann somit kostenlos für beliebige Zwecke vervielfältigt und angepasst werden [36]. 1999 spezifiziert, wurde sie 2001 veröffentlicht [37]. Smarty hat als Template Engine den Vorteil, dass der Entwickler sich auf das Design und die Programmlogik getrennt konzentrieren kann. Die Präsentationsschicht ist von der Programmlogik entkoppelt. Eine Designänderung zieht somit kein komplettes Umschreiben des eigentlichen Programmcodes nach sich, sondern kann allein in der Template-Datei vollzogen werden. Damit die Template-Datei nicht bei jedem Aufruf erneut kompiliert werden muss, werden sogenannte Cache-Files von Templates abgespeichert. Diese erlauben einen schnellen Zugriff auf den eigentlichen HTML Code. Weiter können in den eigentlichen Template-Dateien je nach Anforderung auch PHP Funktionen, Schleifen, Bedingungen, etc. verwendet werden [38]. Listing 4.1 zeigt beispielhaft eine <select> Dropdown Liste mit vorselektiertem Eintrag, die innerhalb einer PHP Datei zusammengebaut wird.

Listing 4.1: Selectbox mit vorselektierter Auswahl in PHP

```
1 $html = '<select␣name="students">';
2 foreach ($students as $student){
3     if ($selectedStudentID == $student['ID']){
4         $html .= '<option␣value="'.$student['ID'].'"␣
              SELECTED␣>'.$student['Name'].'</option>';
```

```
5       } else {
6           $html .= '<option value="'.$student['ID'].'">'.
                $student['Name'].'</option>';
7       }
8   }
9   $html .= '</select>';
```

In Listing 4.2 und 4.3 wurde als Vergleich dieselbe Funktionalität mit Smarty umgesetzt. Dabei ist dank des Smarty Template Aufrufs die Datenschicht von der Präsentationsschicht getrennt. Daten werden an Smarty übergeben, die in der Template-Datei weiterverarbeitet werden können.

Listing 4.2: PHP Code mit Variablen-Zuweisungen an Smarty

```
1 $smarty->assign('selectedStudentID', $selectedStudentID);
2 $smarty->assign('options', $students);
3 $smarty->display('example.tpl');
```

Listing 4.3: Smarty Template Code

```
1 <select name="students">
2     {html_options options=$options selected=
          $selectedStudentID}
3 </select>
```

In Templates selbst können weitere Template-Dateien inkludiert werden, oder durch Plugins eigene PHP Funktionen aufgerufen und Parameter übergeben werden. Diese werden unter dem Dateipfad *smarty/plugins* abgelegt und können mit ihrem Dateinamen im Template aufgerufen werden [38]. Listing 4.4 zeigt die Parameterübergabe an ein Plugin, das in diesem Fall unter *smarty/plugins/function.studentenuebersicht.php* aufgerufen wird. In dieser PHP Datei kann wiederum PHP Code ausgeführt werden, um die eigentliche Programmlogik von der Template Datei fernzuhalten. Beispielweise könnte hier eine komplexe Tabellenstruktur für eine Studentenübersicht in PHP Logik realisiert werden.

Listing 4.4: Smarty Plugin Code

```
1 {foreach from=$studenten item=student}
2     {studentenuebersicht student=$student}
3 {/foreach}
```

4.1.4 TCPDF

In der Entwickler-Dokumentation des Blauen Systems wird irrtümlich FPDF als PDF-Library angegeben [39]. Allerdings handelt es sich bei der verwendeten Library um TCPDF. Sie ist eine Erweiterung, beziehungsweise Verbesserung der FPDF-Library, die mittlerweile nicht mehr weiterentwickelt wird. Im Jahr 2005 [40] erstmals veröffentlicht und mit einer guten Dokumentation ausgestattet, ist sie ebenfalls unter LGPL lizenziert und kann somit frei verwendet werden [36]. Für die Entwicklung mit TCPDF werden für die Basisfunktionen keine weiteren externen Libraries verwendet. Sie unterstützt unter anderem UTF-8, zahlreiche Bildformate und Barcodes. Weiter werden automatische Seitennummerierungen, Seiten- und Zeilenumbrüche, als auch Textausrichtungen angeboten [41].
Im Blauen System wird die Bibliothek dafür verwendet, Bescheinigungen und Nachweise zu generieren. Dafür wird zunächst ein PDF Objekt erzeugt. Im Konstruktor kann unter anderem die Orientierung, Maßeinheit, Seitenformat und Kodierung angegeben werden. Anschließend kann die Schriftart definiert werden. Zum Aufbau einer Tabelle kann die *MultiCell* Methode verwendet werden. Ein einfaches Beispiel der Erzeugung einer zweispaltigen Tabelle ist in Listing 4.5 angegeben.

Listing 4.5: Erzeugung einer Tabelle mit TCPDF

```
1 $pdf = new TCPDF('P', 'mm', 'A4');
2 $pdf->SetTitle('Beispiel-Tabelle');
3 $pdf->SetFont('times', 10);
4 $pdf->AddPage('P');
5
6 $pdf->MultiCell(100, 20, 'Beispiel1', 1, 'C', 0, 0);
7 $pdf->MultiCell(100, 20, 'Beispiel2', 1, 'C', 0, 0);
8
9 $pdf->Output('beispiel.pdf', 'D');
```

In Zeile 1 wird dem Konstruktor als Orientierung Hochformat (Portrait), die Maßeinheit in Millimeter und der DIN A4 Seitenmodus übergeben. Zeile 6 definiert Breite, Höhe, den eigentlichen Text, den Rahmen, die Ausrichtung, die Füllung und zuletzt die Position in der Tabelle. Diese Position gibt an, ob die Zelle beispielsweise rechts oder unterhalb eingefügt werden soll. In diesem Fall hat die Tabelle zwei Spalten und eine Zeile.

4.2 Funktionen

Nachdem die grundlegenden Technologien des Blauen Systems vorgestellt wurden, können nun die eigentlichen Funktionen des Systems näher betrachtet werden. Der Überblick über den bereits vorhandenen Funktionsumfang trägt auch zum Verständnis über die neuen Anforderungen bei. Dabei wird nicht jeder Menüpunkt einzeln erklärt, sondern eine Zusammenfassung über bestimmte Themengebiete gegeben.

4.2.1 Studierendenverwaltung

Der Schwerpunkt des Systems liegt in der Verwaltung des Studenten während seiner Studienzeit an der Medizinischen Fakultät der Universität Ulm. Hierfür wird dieser bei der Immatrikulation zunächst in die Datenbank eingepflegt. Dies kann durch den Import von CSV Listen oder durch eine manuelle Eingabe der Daten in einer Eingabemaske geschehen. Diese ist in der Abbildung 4.2 dargestellt.

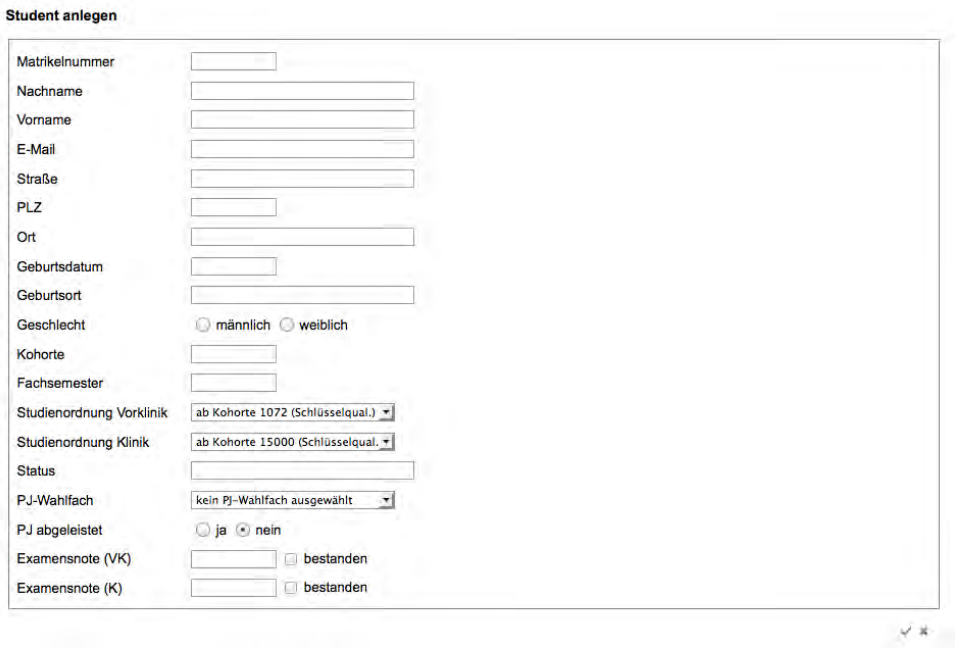


Abbildung 4.2: Eingabemaske zum Anlegen eines neuen Studenten

Eingetragene Studenten können über eine Suchfunktion in der Navigationsleiste gefunden und deren aktueller Studienfortschritt eingesehen und verwaltet werden. Diese Suche ist in Abbildung 4.3 aufgeführt.

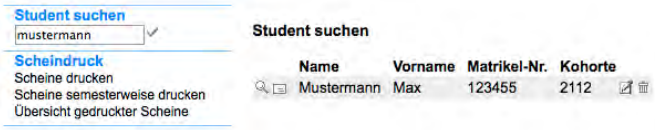


Abbildung 4.3: Suche nach einem Studenten

Übersicht

In den nun folgenden Abschnitten werden die Menüpunkte der Studentendetails kurz erklärt. Dafür wurden in Abbildung 4.4 die entsprechenden Menüpunkte markiert, um anschließend im Text darauf zu verweisen.
In der Abbildung ist die Übersichtsseite, markiert mit der Zahl (1), bereits geöffnet. Hier kann der aktuelle Studienfortschritt auf einen Blick eingesehen werden. Der Studienfortschritt wird hier als Balkengrafik (7) angezeigt. Zudem werden wichtige Studentendaten, als auch die zuletzt vorgenommenen Noteneintragungen angezeigt.

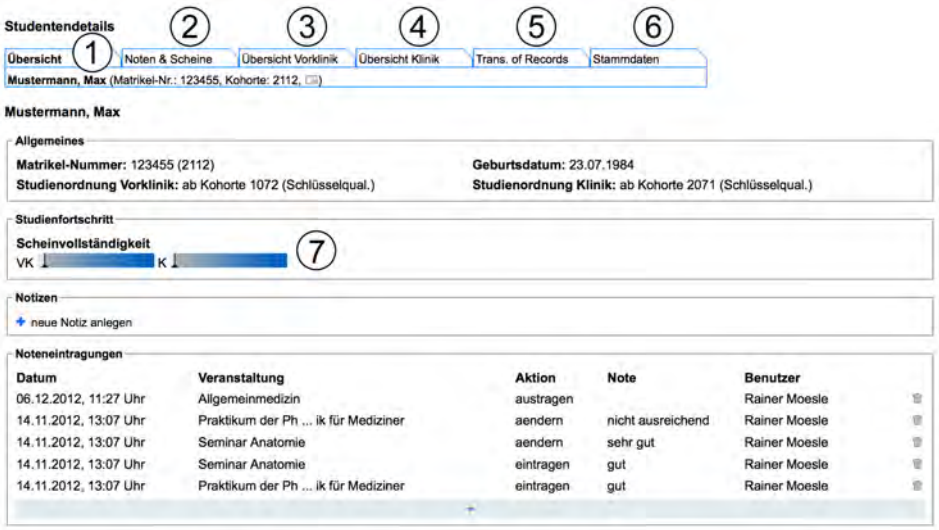


Abbildung 4.4: Die Studentendetails eines eingepflegten Studenten

Noten & Scheine

Im Reiter (2) *Noten & Scheine*, wie in Abbildung 4.5 zu sehen, wird der Status des Praktischen Jahres und alle Veranstaltungen - nach Semestern kategorisiert - angezeigt, für die Prüfungsleistungen vorliegen. Hier kann zum einen für jede Veranstaltung die Benotung geändert, zum anderen auch Nachweise gedruckt werden.

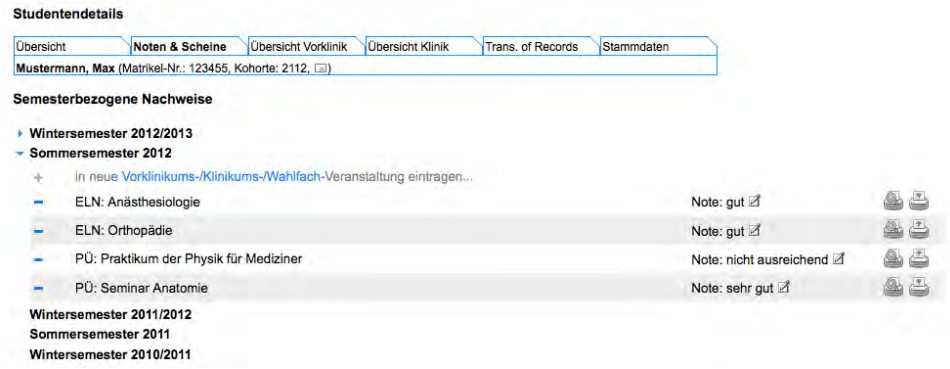


Abbildung 4.5: Die Detailansicht zu Noten & Scheine

Übersicht Vorklinik

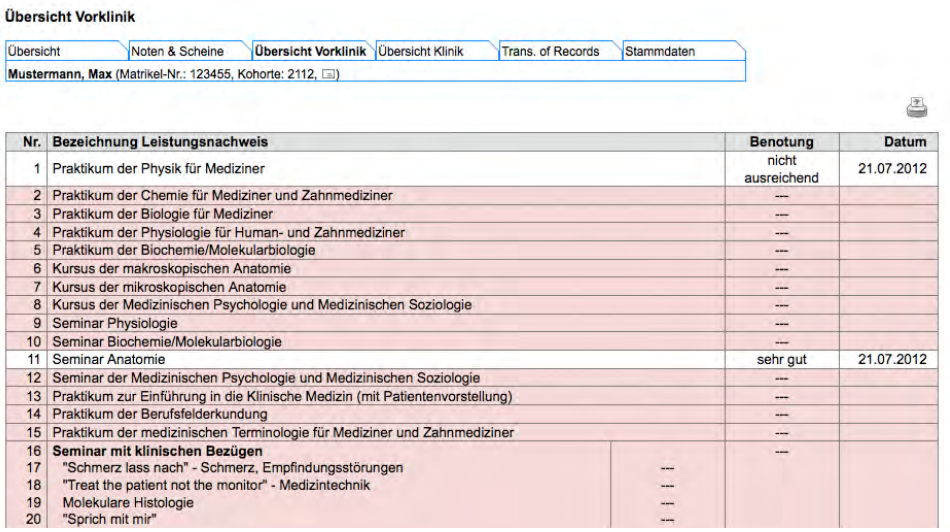
Unter dem Punkt (3) *Übersicht Vorklinik* erhält der Benutzer eine Auflistung der in der Studienordnung zugeordneten Scheine der Vorklinik. Wie bereits in Abschnitt 3.1.4 erklärt, existieren für Klinik und Vorklinik jeweils unterschiedliche Studienordnungen. Diesen Studienordnungen sind immer eine bestimmte Anzahl an Scheinen zugeordnet. In dieser Übersicht werden alle abgelegten Prüfungen inklusive der Note und des Datums angezeigt, noch fehlende Scheine sind rot markiert.

Übersicht Klinik

Dasselbe gilt auch für die *Übersicht Klinik*, hier markiert mit (4), in der die entsprechenden Scheine der Klinik aufgelistet werden. Hier kann ebenfalls, wie auch bei der Vorklinik, eine PDF-Leistungsübersicht über alle abgelegten Leistungen erstellt und ausgedruckt werden. Beispielhaft wurde die Übersicht der Vorklinik in Abbildung 4.6 dargestellt.

Transcript of Records

Dies ist die Übersicht für internationale Studenten, gekennzeichnet mit der Zahl (5). Hier werden die Scheine der Vorklinik und Klinik für internationale Zwecke aufbereitet. Dementsprechend werden, wie in Abbildung 4.7 ersichtlich, auch die internationalen Bezeichnungen und ECTS Creditpoints [6] dargestellt. Auch für diese Übersicht kann ein Leistungsnachweis generiert werden.



Übersicht Vorklinik

Übersicht | Noten & Scheine | **Übersicht Vorklinik** | Übersicht Klinik | Trans. of Records | Stammdaten

Mustermann, Max (Matrikel-Nr.: 123455, Kohorte: 2112,)

Nr.	Bezeichnung Leistungsnachweis		Benotung	Datum
1	Praktikum der Physik für Mediziner		nicht ausreichend	21.07.2012
2	Praktikum der Chemie für Mediziner und Zahnmediziner		---	
3	Praktikum der Biologie für Mediziner		---	
4	Praktikum der Physiologie für Human- und Zahnmediziner		---	
5	Praktikum der Biochemie/Molekularbiologie		---	
6	Kursus der makroskopischen Anatomie		---	
7	Kursus der mikroskopischen Anatomie		---	
8	Kursus der Medizinischen Psychologie und Medizinischen Soziologie		---	
9	Seminar Physiologie		---	
10	Seminar Biochemie/Molekularbiologie		---	
11	Seminar Anatomie		sehr gut	21.07.2012
12	Seminar der Medizinischen Psychologie und Medizinischen Soziologie		---	
13	Praktikum zur Einführung in die Klinische Medizin (mit Patientenvorstellung)		---	
14	Praktikum der Berufsfelderkundung		---	
15	Praktikum der medizinischen Terminologie für Mediziner und Zahnmediziner		---	
16	**Seminar mit klinischen Bezügen**		---	
17	"Schmerz lass nach" - Schmerz, Empfindungsstörungen	---		
18	"Treat the patient not the monitor" - Medizintechnik	---		
19	Molekulare Histologie	---		
20	"Sprich mit mir"	---		

Abbildung 4.6: Die Detailansicht zur Übersicht Vorklinik

Stammdaten

Sollten sich die Anschrift oder sonstigen relevanten Daten eines Studenten während des Studiums ändern, können diese in Reiter (6), den *Stammdaten* des jeweiligen Studenten, bearbeitet werden.

4.2.2 Scheine

Jede Studienordnung, ob Vorklinik oder Klinik, definiert eine Anzahl an ihr zugehörigen Scheinen. Diese Scheine können hierfür im System angelegt und bearbeitet werden. Sie müssen nur einmal angelegt werden. Um das Studium erfolgreich zu beenden, muss ein Student die erfolgreiche Teilnahme an Scheinveranstaltungen vorweisen können. Hierbei werden jedes Semester Veranstaltungen angeboten, die einem bestimmten Schein zugeordnet sind. Legen ein oder mehrere Studenten eine Prüfung in einer Veranstaltung ab, werden diese Noten entweder von Hand in die Datenbank eingetragen oder mit einer CSV Liste importiert. Diese Liste enthält eine Spalte mit der Matrikelnummer der Studenten und deren erreichten Noten. Diese Note variiert von 100 bis 600. Von 100 bis 500 entsprechen diese Noten Schulnoten der Bezeichnung *sehr gut* bis *nicht ausreichend*. Die Zahl 600 bezeichnet eine Veranstaltung als *bestanden*. Diese Note wird anschließend in der Scheinübersicht in Abbildung 4.5 und in der bereits angesprochenen *Übersicht Vorklinik* oder *Übersicht Klinik* des Studenten angezeigt. Für diese Scheine können ebenfalls Bescheinigungen gedruckt werden, sofern hierfür eine Druckvorlage existiert.

Transcript of Records

Übersicht | Noten & Scheine | Übersicht Vorklinik | Übersicht Klinik | **Trans. of Records** | Stammdaten

Mustermann, Max (Matrikel-Nr.: 123455, Kohorte: 2112,)

Preclinical Phase of Studies

Course Name	Course Type	Hours	ECTS Credits	Local Grade
Biology	Lec+Crs	70+40	4	---
Chemistry	Lec+Crs	56+42	7	---
Physics	Lec+Crs	49+42	6	5
Anatomy	Lec+Sem	56+24	5	1
Terminology	Lec+Sem	14+14	2	---
Medical Psychology and Sociology I	Lec+Sem	42+28	3	---
Medical Psychology and Sociology II	Lec+Crs	28+38	3	---
Introduction to Professional Fields	Crs	14	1	---
Introduction to Clinical Medical Professional Fields	Crs	22	3	---
Seminar with Clinical Connections	Sem	56	3	---
Microscopic Anatomy	Lec+Crs	42+52	12	---
Physiology I	Lec+Crs	56+72	11	---
Physiology II	Lec+Sem	56+28	8	---
Gross Anatomy	Lec+Crs	28+114	20	---
Biochemistry, Molecular Biology I	Lec+Crs	70+72	14	---
Biochemistry, Molecular Biology II	Lec+Sem	70+28	9	---
Integrated Clinical Seminars	Sem	98	5	---
Preclinical Elective Subject	Crs	28	2	---
	---	---	5	---

Abbildung 4.7: Die Detailansicht zu Transcript of Records

4.2.3 Veranstaltungen und Wahlfächer

Um zu vermeiden, dass jedes Semester wiederkehrende Veranstaltungen neu eingetragen werden müssen, bietet das System die Möglichkeit, Veranstaltungen aus einem vorangegangen Semester zu kopieren. Dies geschieht, indem der Benutzer beim Anlegen eines neuen Semesters die Möglichkeit auswählt, Veranstaltungen aus einem vorherigen Semester zu kopieren. Generell kann einer Veranstaltung ein Code, eine Gewichtung, ein Schein und ein Dozent zugeordnet werden. Die Gewichtung wird bei Wahlfächern eingetragen und gibt an, wie hoch die Anzahl der Gesamtstunden des Wahlfachs ist. Diese berechnet sich aus den Semesterwochenstunden, kurz SWS. Diese variieren zwischen 1 oder 2 SWS. Daraus ergibt sich eine Gesamtstundenzahl von 14, beziehungsweise 28 Gesamtstunden. Da sich die Note des Wahlfachs in der Leistungsübersicht aus mehreren abgelegten Wahlfächern zusammensetzt, wird die Note anhand der Gewichtung der einzelnen Wahlfächer berechnet. Wie bereits in Abschnitt 4.2.2 angesprochen, wird durch Zuordnung eines Scheins die Note der Veranstaltung in der Scheinübersicht des Studenten (vgl. Abbildung 4.5) angezeigt. Neben Veranstaltungen können auch Wahlfächer für das Praktische Jahr eingetragen und bearbeitet werden. Diese sind nicht zu verwechseln mit den Wahlfächern der Vorklinik und Klinik. Diese werden Veranstaltungen als Scheine zugeordnet. Somit werden Veranstaltungen, die als Vorklinische Wahlfächer eingeordnet werden, dem Schein *Wahlfach*, Klinische Wahlfächern dem Schein *Klinisches Wahlfach*, zugeordnet.

4.2.4 Bescheinigungen, Leistungsübersichten und -nachweise

In vielen Fällen ist es notwendig, Daten aus dem System in Papierform zu exportieren. Dafür gibt es diverse Möglichkeiten, aus bestimmten Daten eine PDF-Datei erzeugen zu lassen. Für teilgenommene Veranstaltungen kann für jede einzelne Prüfungsleistung in der Studentenansicht eine Bescheinigung gedruckt werden, wenn hierfür eine Vorlage existiert. Im ausgewählten Semester können im Menüpunkt *Scheine semesterweise drucken* alle noch nicht erstellten und im Semester gültigen Bescheinigungen eingesehen werden.
Sollen Leistungsnachweise für Vorklinik oder Klinik gedruckt werden, kann dies ebenfalls in der Detailansicht der Studenten im Reiter *Übersicht Vorklinik* und *Übersicht Klinik* erledigt werden. Diese Leistungsübersichten können auch im Menüpunkt *Leistungsübersichten* für ganze Kohorten erstellt werden. In Abbildung 4.8 wurde eine Leistungsübersicht der Vorklinik erstellt. Der Leistungsnachweis des klinischen Studienabschnittes spielt für die Verwaltung eine große Rolle. Daher kann für alle Studenten, die den klinischen Studienabschnitt bestanden haben, automatisch überprüft werden, ob bereits ein Leistungsnachweis ausgedruckt wurde. Diese Möglichkeit findet sich unter *Leistungsnachweis Klinik*. Wurden bereits alle Nachweise erstellt, erscheint ein entsprechender Hinweis.

4.2.5 Praktisches Jahr

Erfüllt ein Student alle Anforderungen, um für das Praktische Jahr zugelassen zu werden, kann der Benutzer dies entweder manuell in den Stammdaten des Studenten erfassen oder durch den Import einer CSV-Liste, die eine Liste an Matrikelnummern enthält.

4.3 Fazit

Nachdem in diesem Kapitel die verwendeten Technologien und die vorhandenen Funktionen vorgestellt wurden, kann im folgenden Kapitel auf die Anforderungen eingegangen werden, die an diese Arbeit gestellt werden. Diese beziehen sich zum einen auf die Erweiterung hier vorgestellter, als auch auf die Umsetzung neuer Funktionen.

ulm university universität **uulm**

Leistungsübersicht - Vorklinischer Abschnitt

Vorname: **Max**
Name: **Mustermann**
Matrikelnummer: **123455 (2112)**
Geburtsdatum: **23.07.1984**

Studiendekanat Medizin
e-mail: med.studiendekanat@uni-ulm.de
Tel.: (0731) 50-22250

Nr.	Bezeichnung Leistungsnachweis		Benotung	Datum
1	Praktikum der Physik für Mediziner		nicht ausreichend	21.07.2012
2	Praktikum der Chemie für Mediziner und Zahnmediziner		---	
3	Praktikum der Biologie für Mediziner		---	
4	Praktikum der Physiologie für Human- und Zahnmediziner		---	
5	Praktikum der Biochemie/Molekularbiologie		---	
6	Kursus der makroskopischen Anatomie		---	
7	Kursus der mikroskopischen Anatomie		---	
8	Kursus der Medizinischen Psychologie und Medizinischen Soziologie		---	
9	Seminar Physiologie		---	
10	Seminar Biochemie/Molekularbiologie		---	
11	Seminar Anatomie		sehr gut	21.07.2012
12	Seminar der Medizinischen Psychologie und Medizinischen Soziologie		---	
13	Praktikum zur Einführung in die Klinische Medizin (mit Patientenvorstellung)		---	
14	Praktikum der Berufsfelderkundung		---	
15	Praktikum der medizinischen Terminologie für Mediziner und Zahnmediziner		---	
16	**Seminar mit klinischen Bezügen**		---	
17	"Schmerz lass nach" - Schmerz, Empfindungsstörungen	---		
18	"Treat the patient not the monitor" - Medizintechnik	---		
19	Molekulare Histologie	---		
20	"Sprich mit mir"	---		
21	**Integrierte Seminare im Umfang von 98 Stunden**		---	
22	"Sport ist Mord" - Sport, Training, Doping, etc.	---		
23	"Mir geht die Luft aus"	---		
24	"Beautiful mind"	---		
25	Pathobiochemie I	---		
26	"Mit 66 Jahren..." - Altern	---		
27	Pathobiochemie II	---		
28	"Deine Gene, Dein Schicksal" - Erbkrankheiten, Gendiagnostik	---		
29	Wahlfach		---	
30	Lernen lernen/Rhetorik/Präsentationstechniken		---	

*) Anerkennung Altschein
**) Anerkennung Ausland
***) Anerkennung Inland

Für die Richtigkeit:

Claudia Grab,
(Referentin für Studium und Lehre)

Ulm, 05.12.2012

Abbildung 4.8: Leistungsübersicht des Vorklinischen Abschnitts

5 Anforderungen

In diesem Kapitel werden die Anforderungen dieser Arbeit vorgestellt. Im nächsten Abschnitt wird zunächst die Zusammenarbeit zwischen der DBIS-Abteilung, dem Studiendekanat der Medizin und dem kiz beschrieben. In Abschnitt 5.2 folgt ein kurze Vorstellung der vom Studiendekanat bereits vor dieser Arbeit definierten Anforderungen. In den darauffolgenden Abschnitten werden die endgültigen Anforderungen an diese Arbeit definiert.

5.1 Zusammenarbeit zwischen DBIS, Medizin und kiz

Grundlage dieser Arbeit war eine enge Zusammenarbeit zwischen dem Institut für Datenbanken und Informationssysteme und dem Studiendekanat der Medizinischen Fakultät. Im Laufe dieser Arbeit fanden zahlreiche Gespräche mit den Mitarbeitern des Studiendekanats statt. Es wurde in ersten Gesprächen auf die Anforderungsvorlage eingegangen und die Machbarkeit der geforderten Erweiterungen diskutiert. Hier fand auch ein Austausch mit dem kiz statt, da zunächst geklärt werden musste, inwieweit ein externer Zugriff auf Corona möglich ist. Corona ist ein Anmeldeverwaltungssystem für Kurs- und Laborplätze. Studenten eines medizinischen Studiengangs können sich online in Corona für teils limitierte Kurse anmelden. Abbildung 5.1 zeigt die Anmeldungsverwaltung aus Sicht eines Mitarbeiters mit Verwaltungsrechten.

Zwar können Mitarbeiter der Medizin über eine Web-Oberfläche Corona-Funktionen nutzen, ein programmatischer Zugriff ist jedoch nur in Zusammenarbeit mit dem kiz möglich. Diese Erkenntnisse flossen in weitere Diskussionsrunden ein, in denen die Anforderungen konkretisiert wurden. Nach einer Einarbeitungsphase wurden erste Funktionen umgesetzt und diskutiert. Fortlaufend mit dem Fortschritt der Arbeit fanden regelmäßig Treffen statt, in denen die Fortschritte diskutiert wurden. Gegen Ende fanden zwei Präsentationen statt, um die Erweiterungen im Gesamtsystem zu präsentieren und um Rückmeldungen zu sammeln.

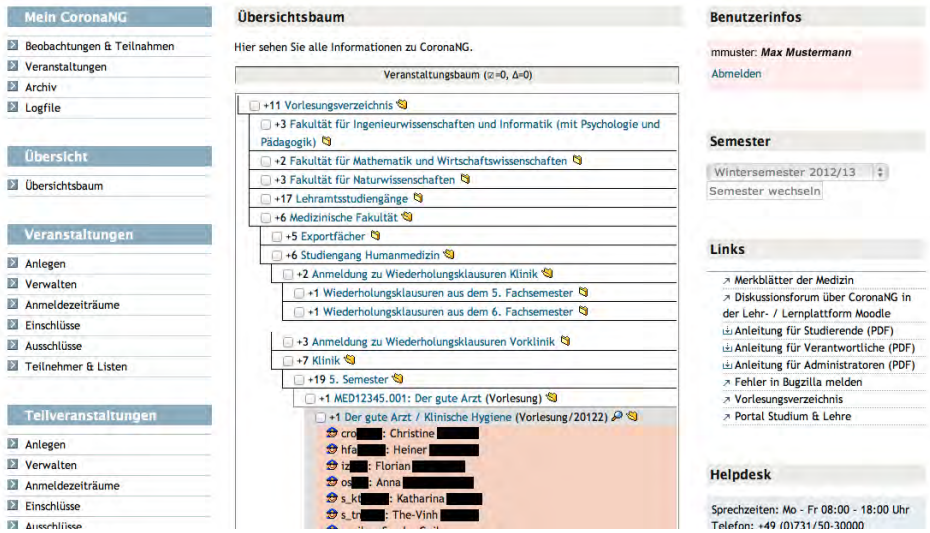


Abbildung 5.1: Die Anmeldungsverwaltung in Corona

5.2 Vorlage des Studiendekanats

Bereits beim ersten Treffen mit Frau Grab und Frau Eichner vom Studiendekanat der Medizin wurde eine Anforderungsliste vorgelegt, datiert auf den 22.06.2011. Abbildung 5.2 zeigt den schematischen Aufbau der Vorlage. Diese Anforderungsliste unterteilt sich in die drei vorhandenen Systeme Moodle, Corona und Blaues System. Die Anforderungen lassen sich grob auf zwei Themen reduzieren, dem Zusammenspiel von Moodle und Corona und dem Zusammenspiel von Corona und dem Blauen System.

5.2.1 Moodle <-> Corona

Das Moodle System, das von der Medizin als Lernplattform genutzt wird, soll in geeigneter Weise mit dem Kurs- und Klausuranmeldesystem Corona interagieren. Im Corona System sind für jeden Studenten die Anmeldungen zu Kursen und Klausuren gespeichert. Diese Daten sollen dazu genutzt werden, jeden Studenten nach der Anmeldung in Corona automatisch für einen Moodle Kurs anzumelden, beziehungsweise diesen Kurs zu abonnieren. Für Klausurwiederholer soll dies auch manuell möglich sein. Des Weiteren soll Corona, als auch das Evaluationssystem EvaSys, in geeigneter Weise mit einer Schnittstelle versehen werden, sodass eine Klausuranmeldung automatisch eine Evaluation in EvaSys nach sich zieht. EvaSys ist eine Software zur Lehrevaluation an der Universität Ulm. Sie ermöglicht die Erstellung und Auswertung von papier- oder webbasierten Fragebögen und Um-

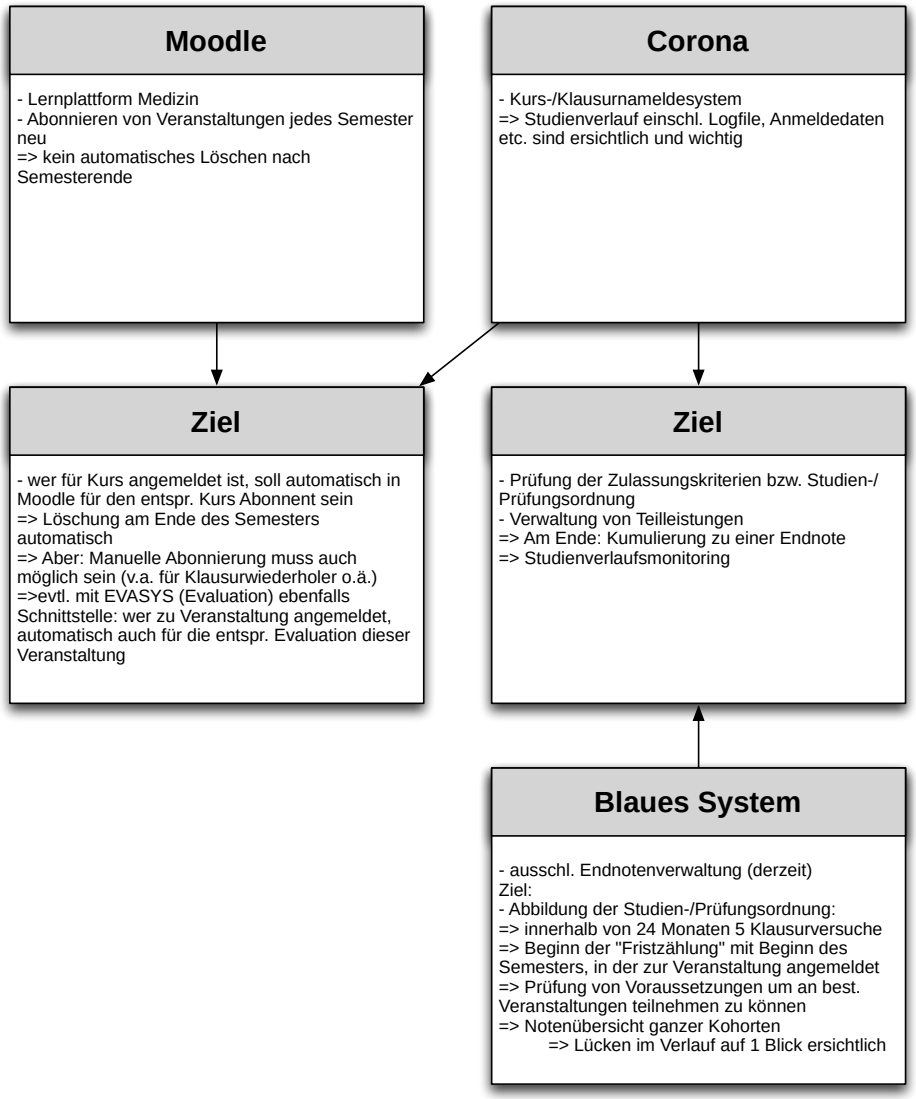


Abbildung 5.2: Die Anforderungsliste zu Beginn der Arbeit

fragen [42]. Abbildung 5.3 zeigt eine Liste der angelegten Teilbereiche aus Sicht des Administrators. Diese Teilbereiche sind in diesem Fall Fakultäten, beziehungsweise zentrale Einrichtung. Bei der Erstellung von Fragebögen können diese den Teilbereichen zugeordnet werden.

5.2.2 Blaues System <-> Corona

Im Blauen System sollen die Studienordnungen in der Art und Weise abgebildet werden, dass eine Plausibilitätsprüfung des Studienverlaufs jedes einzelnen Studenten möglich ist. Dafür muss neben der Erfassung von Klausurfristen auch die

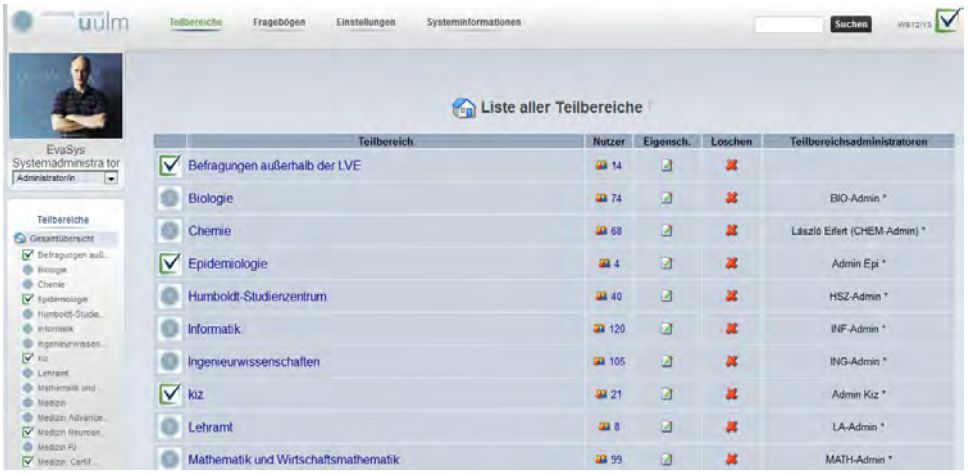


Abbildung 5.3: Übersicht aller Teilbereiche in EvaSys

Erfassung der Klausurversuche möglich sein. Dies kann je nach Studienordnung variieren. Weiter soll das Blaue System in der Plausibilitätsprüfung auch Schein-Voraussetzungen berücksichtigen. Bestimmte Veranstaltungen setzen das erfolgreiche Ablegen weiterer Veranstaltungen voraus. Dies soll konkret im Zusammenspiel mit Corona geschehen. Aus Corona werden die angemeldeten Prüflinge abgefragt und mit den im Blauen System erfassten Prüfungsleistungen abgeglichen. Auf Seiten des Blauen Systems soll zudem eine Leistungsübersicht über bestimmte Kohorten abrufbar sein.

5.3 Blaues System

In diesem Abschnitt werden alle Anforderungen definiert, die auf Seiten des Blauen Systems gefordert sind. Diese decken sich größtenteils mit der Vorlage aus Abschnitt 5.2, sind aber durch kontinuierliche Gesprächsrunden und Rückmeldungen seitens des Studiendekanats detailreicher ausgeführt. Für jeden Anforderungsbereich wird zudem eine chronologische Übersicht darüber gegeben, wie die Anforderungen umgesetzt wurden, beziehungsweise sich im Verlauf dieser Arbeit geändert haben.

5.3.1 Teilleistungen

Bisher erhielt das Studiendekanat von den medizinischen Abteilungen sogenannte Endnoten, die anschließend in das Blaue System eingetragen wurden. Die Verwaltung der Zwischennoten oblag somit ausschließlich den Abteilungen und konn-

te vom Dekanat nicht überprüft, beziehungsweise nachvollzogen werden. Diesem Misstand soll Abhilfe geschaffen werden, indem es möglich ist, für Veranstaltungen eines Semesters auch Teilleistungen zu definieren. Für jede Teilleistung soll ein Name als Bezeichnung definierbar sein. Außerdem muss es möglich sein, die verschiedenen Arten von Teilleistungen abzubilden. Hier sind nicht nur Noten, sondern auch Punkte vorgesehen, die letztendlich eine andere Berechnung der Endnote nach sich ziehen. Im Falle der Notenvergabe wird aus den eingetragenen Noten durch das arithmetische Mittel eine Endnote gebildet. Eine Aufrundung der Endnote findet ab der Nachstelle ,51 statt. Des Weiteren können auch Punkte vergeben werden. Die Endnote wird hier kumuliert, das heißt die Punkte werden zu einer Gesamtpunktzahl addiert. Laut Studienordnung liegt der Prozentsatz der zu erreichenden Punkte bei 60%. Statt diesen Wertes kann auch eine Gewichtung angegeben werden. Hier kann festgelegt werden, welche Gesamtpunktzahl stufenweise einer schulischen Note entspricht.

Chronologische Umsetzung der Anforderungen

- Anlegen und Bearbeiten von Teilleistungen für Veranstaltungen, abhängig vom ausgewählten Semester.
- Import von Teilleistungen, vergleichbar mit bestehendem Notenimport. Statt Noten werden Prozentpunkte importiert.
- Anzeige von Teilleistungen in Studenten- und Notenansicht.
- Verwerfen der Prozentpunkte. Stattdessen Import von Noten.
- Beim Import Angabe von Prüfungsdatum.
- Berücksichtigung von Wiederholungsprüfungen.
- Bearbeiten und Löschen von Teilleistungen in der Notenansicht.
- Kumulieren einer Endnote beim Import von Teilleistungen.
- Anlegen von Namen für Teilleistungen.
- Konzept überarbeiten. Import von Noten und Punkten.
- Endnoten werden entweder über arithmetisches Mittel gebildet oder kumuliert.
- Berechnung der Endnote überarbeiten.

Beispiel

In Abbildung 5.4 ist beispielhaft die Bearbeitung von Teilleistungen für die Veranstaltung der Allgemeinmedizin dargestellt. Hier kann sowohl die Anzahl der Teilleistungen festgelegt werden, als auch den einzelnen Teilleistungen Eigennamen vergeben werden. Außerdem kann die Berechnung der Endnote festgelegt werden. In diesem Fall wird die Endnote aus den Teilleistungen kumuliert. Ein Wert größer gleich 60% wertet die Prüfung als *bestanden*.

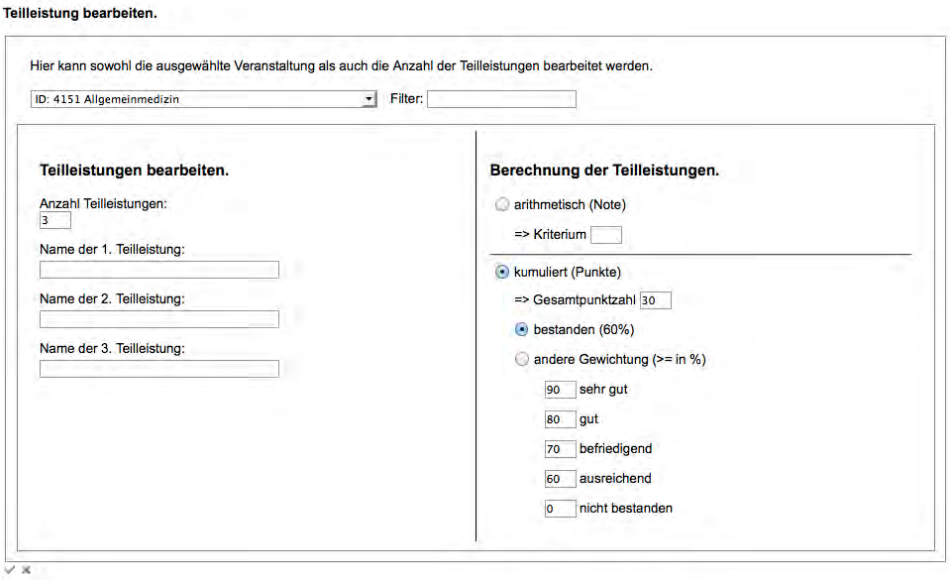


Abbildung 5.4: Bearbeiten einer Teilleistung

5.3.2 Studienordnung

Jedem Studenten wird für die Vorklinik, als auch Klinik, eine Studienordnung zugewiesen. Allerdings wurden diese Studienordnungen bei der Entwicklung des Systems manuell in der Datenbank eingetragen und konnten seither nicht mehr aktualisiert oder erneuert werden. Es soll daher möglich sein, Studienordnungen neu zu erstellen und vorhandene zu bearbeiten. Bezugnehmend auf die Plausibilitäts-Checks im folgenden Abschnitt, müssen zudem für jede Studienordnung weitere Kriterien wie Klausurversuche, Fristen oder Voraussetzungen definierbar sein.

Chronologische Umsetzung der Anforderungen

- Erweiterung der Studienordnung um Fristen und Klausurversuche.

- Definition von Schein-Voraussetzungen.
- Anlegen, Kopieren und Bearbeiten von Studienordnungen.
- Flexible Sortierung bei der Definition von neuen Studienordnung-Scheinen.

Beispiel

Abbildung 5.5 zeigt die Zuordnung eines Scheins zu einer Studienordnung. Dieser Schein war bisher noch nicht zugeordnet und wurde aus einer Auswahlliste ausgewählt. Nun können die Sortierung, Oberschein, Scheintyp, Scheinvorlage und ECTS Punkte zugeordnet werden. Zur Übersicht wird oberhalb dieser Eingabemaske eine Tabelle der bereits zugeordneten Scheine angezeigt.

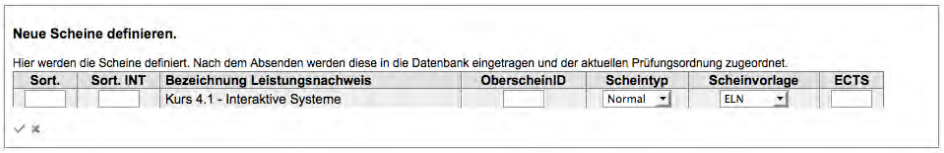


Abbildung 5.5: Zuordnung eines neuen Scheins

5.3.3 Plausibilitäts-Checks

Den Plausibilitäts-Checks kommt eine besondere Bedeutung zu. Bisher wurde diese Prüfung von Mitarbeitern des Studiendekanats manuell anhand von Listen durchgeführt. Im Blauen System war bisher nicht vorgesehen, Auswertungen, beziehungsweise Aussagen über den Verlauf eines Studenten zu treffen. Es ist gefordert, dass Studenten auf die Einhaltung der in der Studienordnung definierten Kritierien überprüft werden können. Hier sind zunächst zwei Kriterien von entscheidender Bedeutung: die Anzahl der Prüfungsversuche und die Einhaltung der Klausurfristen. Je nach Studienordnung sind maximal 5, beziehungsweise in absehbarer Zeit 3 Prüfungsteilnahmen an derselben Veranstaltung vorgesehen. Überschreitet ein Student diese Anzahl an Prüfungsversuchen, soll dies vom System automatisch ermittelt werden. Des Weiteren muss ein Student eine Veranstaltung nach der ersten Prüfungsteilnahme innerhalb von 24 Monaten bestehen. Weiter setzen laut Studienordnung einige Veranstaltungen die erfolgreiche Teilnahme an weiteren Veranstaltungen voraus. Dies soll im Zusammenspiel mit Corona anhand aktueller Klausuranmeldungen überprüft werden. Dabei sollen in Corona verfügbare Veranstaltungen zunächst auf definierte Voraussetzungen überprüft werden. Wurde für eine Scheinveranstaltung eine oder mehrere Voraussetzungen definiert,

werden aus Corona alle angemeldeten Studenten abgefragt. Diese werden dann einzeln mit dem Datenbestand des Blauen Systems auf die Erfüllung der Voraussetzungen überprüft.

Chronologische Umsetzung der Anforderungen

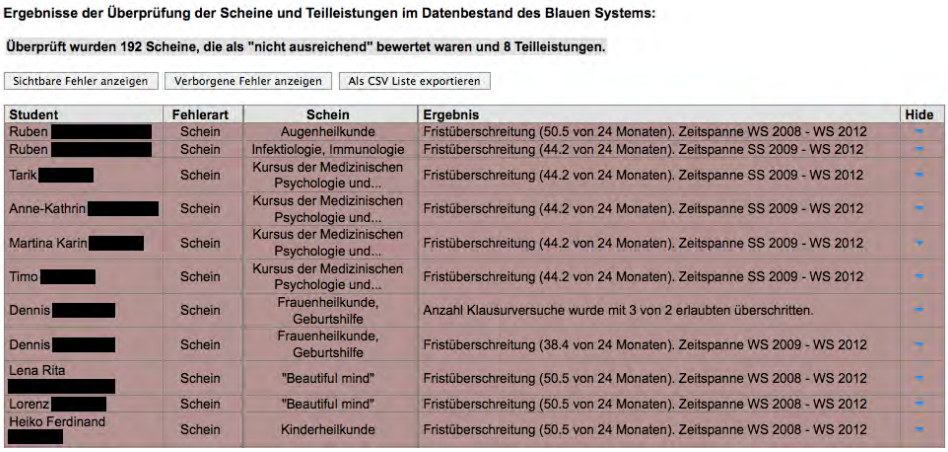
- Abbildung von Klausurversuchen und -fristen je nach Studienordnung.
- Teilleistungen anhand des Vorlesungsanfangs des zugeordneten Scheins prüfbar machen.
- Import von Veranstaltungen inkl. angemeldeter Studenten aus Corona.
- Anhand Matrikelnummerliste mit Datenbestand des Blauen Systems auf Frist-/Klausurversuch-Überschreitungen prüfen.
- Kurz-Check realisieren, der die Anzahl an gefundenen Fehlern anzeigt.
- Klick auf Kurz-Check -> Schnellansicht für Fehler
- Kurz-Checks verwerfen, stattdessen Plausiblitäts-Checks bei Login automatisch starten und im Hintergrund ausführen. Kurzansicht bleibt.
- Verbergen von Fehlern mit Begründung.
- CSV-Export für Fehlerliste.

Beispiel

In Abbildung 5.6 ist ein Ausschnitt des Ergebnisses des Plausibilitäts-Checks dargestellt. Dieser wurde anhand des Datenbestandes des Blauen Systems durchgeführt und auf Klausurversuch- und Fristüberschreitungen der eingetragenen Prüfungsleistungen von Scheinen und Teilleistungen geprüft.

5.3.4 Kohortenansicht

Bisher war es nicht möglich, eingetragene Prüfungsergebnisse für mehr als einen Studenten gleichzeitig aufzurufen. Lediglich für jeden einzelnen Studenten wurden dessen abgelegte Leistungen in einer Übersicht angezeigt. Daher soll es möglich sein, eine Übersicht über jede Veranstaltung eines ausgewählten Semesters zu erhalten. Bei der Auswahl einer Veranstaltung sollen die daran teilgenommenen Kohorten an- und abwählbar sein. Wird eine Kohorte ausgewählt, werden alle



Ergebnisse der Überprüfung der Scheine und Teilleistungen im Datenbestand des Blauen Systems:

Überprüft wurden 192 Scheine, die als "nicht ausreichend" bewertet waren und 8 Teilleistungen.

Sichtbare Fehler anzeigen | Verborgene Fehler anzeigen | Als CSV Liste exportieren

Student	Fehlerart	Schein	Ergebnis	Hide
Ruben [redacted]	Schein	Augenheilkunde	Fristüberschreitung (50.5 von 24 Monaten). Zeitspanne WS 2008 - WS 2012	-
Ruben [redacted]	Schein	Infektiologie, Immunologie	Fristüberschreitung (44.2 von 24 Monaten). Zeitspanne SS 2009 - WS 2012	-
Tarik [redacted]	Schein	Kursus der Medizinischen Psychologie und...	Fristüberschreitung (44.2 von 24 Monaten). Zeitspanne SS 2009 - WS 2012	-
Anne-Kathrin [redacted]	Schein	Kursus der Medizinischen Psychologie und...	Fristüberschreitung (44.2 von 24 Monaten). Zeitspanne SS 2009 - WS 2012	-
Martina Karin [redacted]	Schein	Kursus der Medizinischen Psychologie und...	Fristüberschreitung (44.2 von 24 Monaten). Zeitspanne SS 2009 - WS 2012	-
Timo [redacted]	Schein	Kursus der Medizinischen Psychologie und...	Fristüberschreitung (44.2 von 24 Monaten). Zeitspanne SS 2009 - WS 2012	-
Dennis [redacted]	Schein	Frauenheilkunde, Geburtshilfe	Anzahl Klausurversuche wurde mit 3 von 2 erlaubten überschritten.	-
Dennis [redacted]	Schein	Frauenheilkunde, Geburtshilfe	Fristüberschreitung (38.4 von 24 Monaten). Zeitspanne WS 2009 - WS 2012	-
Lena Rita [redacted]	Schein	"Beautiful mind"	Fristüberschreitung (50.5 von 24 Monaten). Zeitspanne WS 2008 - WS 2012	-
Lorenz [redacted]	Schein	"Beautiful mind"	Fristüberschreitung (50.5 von 24 Monaten). Zeitspanne WS 2008 - WS 2012	-
Heiko Ferdinand [redacted]	Schein	Kinderheilkunde	Fristüberschreitung (50.5 von 24 Monaten). Zeitspanne WS 2008 - WS 2012	-

Abbildung 5.6: Ergebnis des Plausibilitäts-Checks

Studenten der Kohorte angezeigt, die entweder an dieser Veranstaltung, an einer früheren Veranstaltung dieser Art oder noch gar nicht teilgenommen haben. Vorhandene Noten, als auch noch nicht erfasste Leistungen, sollen direkt in der Ansicht editierbar sein.

Chronologische Umsetzung der Anforderungen

- Kohorten in Auswahlliste anzeigen. Für jede Kohorte eine Liste von Studenten und teilgenommenen Veranstaltungen ausgeben.
- Gruppierung von Veranstaltungen für Praktika, Kurse und Seminare.
- Veranstaltungstitel abkürzbar machen (Chemie (P), Biologie (P)), damit mehr Spalten pro Seite angezeigt werden können.
- Student anklickbar -> Einzelansicht der abgelegten Veranstaltungen
- Suche anhand des Namens.
- Paging mit Seitenzahlen, kompletter Veranstaltungstitel per MouseOver.
- Kohortenansicht verwerfen. Nun nach Veranstaltungen und nicht nach Kohorten sortieren.
- Prüfungen direkt editierbar in Ansicht mit HTML Anker.
- Checkbox zur Einschränkung/Anzeige von Kohorten.

Beispiel

Als anschauliches Beispiel der Kohortenansicht dient Abbildung 5.7. Hier wurde für den Klinischen Untersuchungs-Kurs des Sommersemesters 2003 die Kohorte 231 ausgewählt. Für weitere Kohorten liegen hier keine Prüfungsleistungen vor. In der Übersichtstabelle werden zuerst die Studenten angezeigt, für die Ergebnisse für das ausgewählte Semester vorliegen - hier ein einzelner Student. Diese sind nach Nachnamen sortiert. Darunter befinden sich die Studenten, die bereits an einer solchen Scheinprüfung teilgenommen haben - hier im Sommersemester 2005. Anschließend werden die Studenten aufgelistet, für die noch keine Prüfungsergebnisse vorliegen.

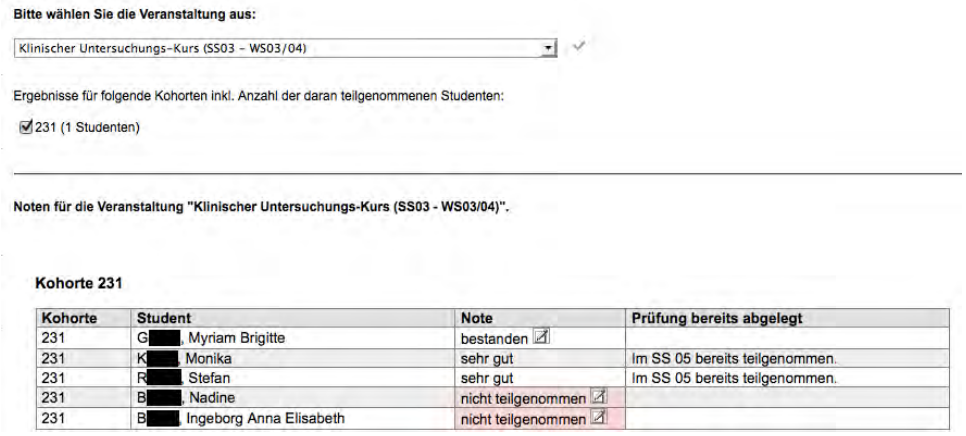


Abbildung 5.7: Kohortenansicht für eine ausgewählte Veranstaltung

5.3.5 Studentenansicht

Eingetragene Teilleistungen sollen in der *Übersicht Vorklinik* und *Übersicht Klinik* zusätzlich zu den Schein-Noten angezeigt werden. Die Studentenansicht soll auf der Übersichtsseite, ähnlich der Noten-Logeintragung, auch die Eintragung für Teilleistungen anzeigen. Des Weiteren soll eine Kurzanzeige der Plausibilitäts-Ergebnisse des ausgewählten Studenten einen schnellen Überblick über konkrete Probleme im Studienverlauf geben.

Chronologische Umsetzung der Anforderungen

- Druck von Bescheinigungen für alle vorhandenen Wahlfächer.
- Anzeige von Ergebnissen aus dem Plausibilitäts-Check auf der Startseite der Studentenansicht.

- Anzeige der Logs für Teilleistungen.

Beispiel

Abbildung 5.8 zeigt ein Ausschnitt der Studentenansicht. Hier wurde eine Teilleistung mit der Note *gut* eingetragen. Für den Studenten liegen bisher keine Ergebnisse des Plausibilitäts-Checks vor.

Abbildung 5.8: Log-Einträge in der Studentenansicht

5.3.6 PDFs

Bei der Generierung von Bescheinigungen und Nachweisen sollen einige statische Inhalte editierbar werden. Zwar werden bisher alle Daten des Studenten dynamisch aus der Datenbank gelesen und in die PDF-Ansicht eingefügt, allerdings sind beispielsweise die Kontaktdaten des Studiendekanats statisch und nicht veränderbar. Ändert sich die Telefonnummer oder der Ansprechpartner, soll es möglich sein, diese Daten frei editieren zu können. Die Möglichkeit der Editierbarkeit ist ebenso für weitere statische Inhalte bei der PDF-Generierung gefordert.
Da zur bisherigen Eintragung von Noten nun auch das Eintragen von Teilleistungen möglich ist, muss auch hierfür eine Erstellung eines Nachweises, ähnlich dem für Noten, möglich sein.

Chronologische Umsetzung der Anforderungen

- Daten des Ansprechpartners in Bescheinigungen und Nachweisen editierbar machen.
- Drucken von Bescheinigungen für Vorklinik und Klinik in einer PDF.
- Drucken von Teilleistungen in der Notenansicht.
- Neue Leistungsübersicht für die Anerkennung von Studienleistungen.
- Drucken von Bescheinigungen für Vorklinik und Klinik in einer PDF für internationale Studenten (TOR).
- Info-Text in der Bescheinigung des Praktischen Jahres editierbar machen.

Beispiel

Die Editierfunktion für die Informationen, die bei der PDF-Generierung automatisch aus der Datenbank geladen werden, ist in Abbildung 5.9 dargestellt. Hier ist ein kleiner Ausschnitt der Eingabefelder abgebildet, mit denen die Daten der Fakultät, des Ansprechpartners im Studiendekanat und weitere Infotexte in den PDF-Bescheinigungen und -Nachweisen geändert werden können.

Abbildung 5.9: Bearbeiten der PDF Info-Daten

5.4 Corona

Um die in Abschnitt 5.3.3 angesprochenen Plausibiltätschecks durchführen zu können, muss auf Seiten Coronas eine Schnittstelle geschaffen werden. Diese muss vom Blauen System aus ansprechbar sein und zwei Funktionen bereitstellen. Im ersten Schritt muss aus Corona eine Liste aller relevanten Veranstaltungen eines Semesters exportierbar sein. Im zweiten Schritt muss anhand einer dieser Veranstaltungen eine Liste an teilnehmenden Studenten zurückgegeben werden.

Chronologische Umsetzung der Anforderungen

- WebService-Aufruf testen (auth-WebService).
- Veranstaltungen jedes Semesters aus Corona in Blaues System importieren. Flag und Importdatum in DB um importierte und manuell angelegte Veranstaltungen zu unterscheiden.
- Manuelles Eintragen von LSFIDs in nicht-importierte Veranstaltungen.
- Import-Kriterien für den Import aus Corona definieren, da sich LSFIDs unterscheiden können.
- Kriterien für die Zuordnung von Corona-Veranstaltungen -> Scheine aus Blaues System. Titel sollen immer gleich bleiben.

- Studiendekanat kontaktieren, um aktuellen Datenbank Dump zu erhalten.
- Veranstaltungs-Import verwerfen. Veranstaltungen sollen wie bisher semesterweise mit Bordmitteln des Blauen Systems kopiert werden.
- ImportKriterien nur noch für Plausibilitäts-Checks.

5.5 MedicUlm

Um auch für Studenten einen Mehrwert zu bieten, soll eine Notenabfrage durch die bereits bestehende *MedicUlm* Android App [43] der Medizinischen Fakultät möglich sein. Der Benutzer soll sich über ein Loginverfahren mit seinem kiz-Login und -Passwort authentifizieren und anschließend eine Notenansicht erhalten. Diese Abfrage soll allerdings nicht manipulierbar sein und der Server des Blauen Systems nach außen anonym bleiben, sodass die App über eine Zwischenstelle mit dem Blauen System kommuniziert. Ein WebService soll hierbei die Kommunikation zwischen MedicUlm und Blauem System leiten, sodass das System aus Sicht des Anwenders im Hintergrund bleibt.

Chronologische Umsetzung der Anforderungen

- Einarbeitung in vorhandene Android App.
- Menüpunkt und Login-Screen zur Eingabe von Benutzername und Passwort.
- Parsen und Anzeige der XML-Notenliste.
- Auswahlmöglichkeit für Vorklinik/Klinik.

Beispiel

Die Umsetzung der Anforderungen an die MedicUlm App ist in Abbildung 5.10 dargestellt. Nach Eingabe der kiz Login-Daten zeigt die Android App eine Notenübersicht in Form einer Liste an. Zwischen Vorklinik und Klinik kann mithilfe einer Auswahlliste gewählt werden.

5.6 WebService

Es soll ein WebService realisiert werden, der mit MedicUlm und dem Blauem System kommuniziert. Dieser soll die Anmeldedaten des Benutzers an das Blaue

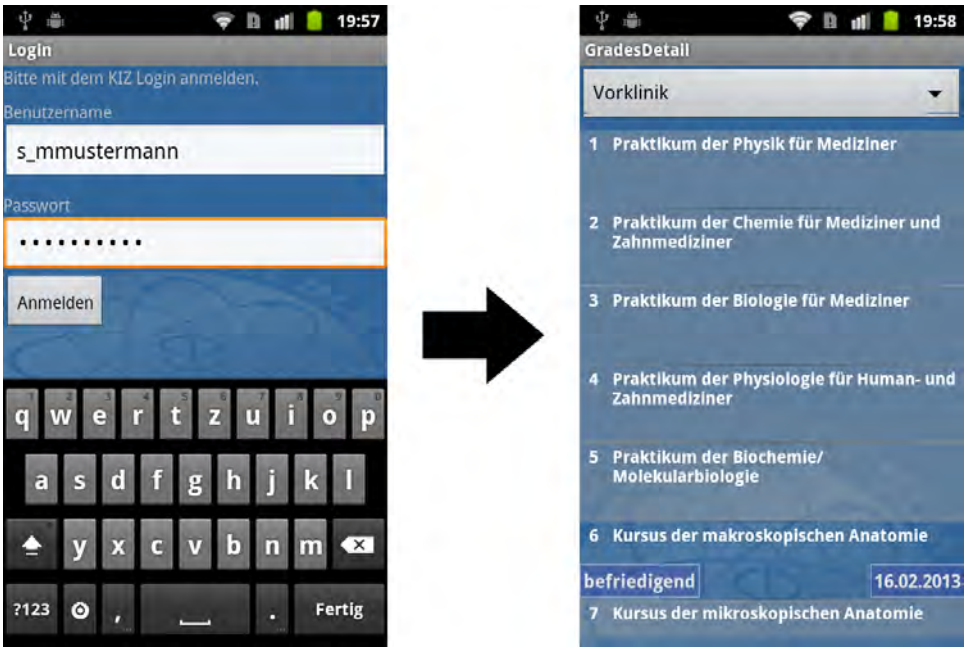


Abbildung 5.10: Login und Notenansicht in der MedicUlm App

System weiterreichen. Als Antwort erhält der WebService eine XML-Liste der Prüfungsleistungen des Studenten oder im Fehlerfall eine Mitteilung, dass die Logindaten nicht verifiziert werden konnten. Diese Nachrichten reicht er dann wiederum an die App weiter. Die Verifizierung der Logindaten nimmt das Blaue System vor, indem dies wiederum einen WebService aufruft und die Anmeldedaten übergibt. Dieser WebService überprüft die Daten über den LDAP Server der Universität Ulm.

Chronologische Umsetzung der Anforderungen

- getGrades WebService: Generierung einer Zeichensequenz, die an das Blaue System und an App weitergegeben wird. Zeichensequenz und Matrikelnummer im Blauem System speichern.
- Realisierung einer LDAP-Schnittstelle auf PHP-Basis.
- Verwerfen des Konzeptes. Stattdessen Benutzername und Passwort an Blaues System weiterreichen. Blaues System nutzt vorhandenen auth-WebService der DBIS zur Überprüfung des Logins. WebService bekommt Notenliste von Blauem System und gibt diese in der Response zurück.

5.7 Moodle

Moodle ist eine an der Universität Ulm verwendete E-Learning Software, die vom kiz seit 2011 bereitgestellt wird. Über Moodle können unter anderem Lerninhalte, virtuelle Seminarräume oder Foren bereitgestellt werden. Der Zugang hierzu kann auch beschränkt werden [44]. Die Lernplattform soll so erweitert werden, dass es möglich ist, einen Studenten, der sich in Corona für eine Veranstaltung anmeldet, automatisch für einen entsprechenden Moodle Kurs zu registrieren. Dabei soll es aber weiterhin möglich sein, Kurse selbst zu abonnieren, beispielsweise, wenn eine Prüfung wiederholt wird.

5.8 Schwierigkeiten

Im Laufe dieser Arbeit konnten wichtige Erkenntnisse für die Praxis gewonnen werden. Bei einem Projekt kann es an verschiedenen Stellen Schwierigkeiten geben, sei es in der Kommunikation, der Planung oder auch in der Ausführung. Im folgenden wird auf diese Probleme eingegangen.

5.8.1 Einarbeitung

Bevor konkrete Gespräche über Erweiterungen des Blauen Systems geführt werden konnten, war es nötig, sich mit dem System und den Details eines Medizinstudiums vertraut zu machen. Hier gab es zunächst Schwierigkeiten bezüglich des Datenbank-Dumps, da dieser über PhpMyAdmin nicht exportierbar war und nur über einen Dump der Konsole bewerkstelligt werden konnte. Die Ursache dafür ist möglicherweise auf die Konvertierung der Tabellen zu UTF-8, beziehungsweise einer nachträglichen Konvertierung von MyISAM zu InnoDB zurückzuführen. Der Dump konnte zunächst nicht importiert werden, da die Erzeugungs-Reihenfolge der Tabellen zunächst manuell umgestellt werden musste. Für das Blaue System selbst existiert keine ausreichende Dokumentation. Die knapp bemessenen Entwicklernotizen erinnern eher an Kurznotizen für die vorherigen Entwickler, als dass sie für die Erweiterung des Systems von Relevanz wären. Einige der Funktionalitäten sind zwar auch ohne Kommentierung selbsterklärend, andere wiederum würden unbedingt eine ausführlichere Dokumentation erfordern. Ein Beispiel hierfür wäre die Funktion zum Druck von Nachweisen, im konkreten Fall eines Scheins. Über verschiedene Abstraktionsstufen hinweg werden Funktionen aus verschiedenen Dateien ohne jegliche Kommentierung der Zusammenhänge aufgerufen. So-

mit erweist sich die Einarbeitung als äußerst mühselig, um die einzelnen Schritte nachzuvollziehen und die jeweilige Vorgehensweise in ihrer Gesamtheit zu verstehen. Aus diesem Grund werden im Implementierungs-Kapitel im Abschnitt 7.5 verschiedene Funktionen erläutert, deren Zusammenhänge nicht ohne weiteres verständlich sind.

5.8.2 Gespräche

Es wurde in den Gesprächen deutlich, dass es in der Praxis schwierig ist, die fachlichen Anforderungen der Dekanatsmitarbeiter auf technische Programmlogik umzumünzen und Lösungswege aufzuzeigen. Beschränkungen des vorhandenen Systems müssen für Mitarbeiter ohne technisches Hintergrundwissen aufbereitet und in für sie bekannte Begrifflichkeiten veranschaulicht werden. Zudem verfügen die Mitarbeiter über unterschiedliche Sichtweisen und haben je nach Aufgabengebiet mehr oder weniger Erfahrung mit dem System. Somit muss in den Gesprächen, beziehungsweise Vorträgen, der Kenntnisstand jedes einzelnen Mitarbeiters berücksichtigt werden, um ihn adäquat in die Anforderungsanalyse einzubinden.

5.8.3 Implementierung

Weitere Schwierigkeiten tauchten während der Implementierung der Anforderungen auf. Durch Präsentationen von Zwischenständen konnten sich die Mitarbeiter ein genaueres Bild der anprogrammierten Funktionen machen und Anmerkungen einfließen lassen. Hier fand ein stetiger Austausch statt, der zum einen die Wünsche der Mitarbeiter direkt einbeziehen, zum anderen unnötigen Programmieraufwand reduzieren sollte. Bei der ein oder anderen Funktion wurde allerdings erst in der Präsentation für einen erweiterten Mitarbeiterkreis deutlich, dass zum Beispiel die Kohortenübersicht oder die Teilleistungen entweder komplett unbrauchbar waren oder zuerst Informationen von anderen Abteilungen eingeholt werden müssen. Zuvor war man sich im kleinen Kreis mit den direkten Ansprechpartnern über die Details einig gewesen. Dies lässt sich am anschaulichsten an der Kohortenübersicht aufzeigen. In Kleingruppengesprächen einigte man sich zunächst darauf, die Anzeige von Prüfungsergebnissen für einzelne Kohorten zu realisieren. Es war gewünscht, eine Kohorte über eine Auswahlliste selektieren zu können und in einer tabellarischen Ansicht die Prüfungsergebnisse jedes Studenten dieser Kohorte einzusehen. Hierfür wurden in den Spalten jeweils vier Veranstaltungen gleichzeitig angezeigt, darunter in den Zeilen jeweils das Ergebnis des jeweiligen Studenten, ob er an dieser Veranstaltung teilgenommen hat oder nicht. Weitere Veranstaltun-

gen waren über eine Paging Funktion zugänglich. Was sich allerdings im Laufe dieser Arbeit durch eine Präsentation mit allen Mitarbeitern herausstellte, war, dass diese Funktionen keinen Mehrwert bietet. Vielmehr sollte der Schwerpunkt nicht auf den Kohorten selbst, sondern auf den Veranstaltungen liegen. Es soll vielmehr pro Veranstaltung eines Semesters möglich sein, die Ergebnisse aller Studenten anzuzeigen, nicht nur für eine Kohorte. Dadurch lassen sich auf einen Blick die Studenten extrahieren, die bisher noch keinen Prüfungserfolg vorweisen können. Somit war ein komplettes Neuschreiben der eigentlich bereits abgeschlossenen Funktion unumgänglich. Als weiteres Beispiel sei die Erweiterung der Funktionalität um Teilleistungen angeführt. Hier einigte man sich auf die reine Eintragung von Noten. Aus diesen Noten sollte für die Endnote der Veranstaltung, beziehungsweise des Scheins, eine Endnote über das arithmetische Mittel gebildet werden. Auch diese Funktion war bereits fast fertig implementiert, bis in einer abschließenden Präsentation im erweiterten Mitarbeiterkreis klargestellt wurde, dass nicht nur Noten, sondern auch Punkte und eventuell weitere Leistungsarten erfasst werden müssen. Hierfür mussten allerdings erst andere Abteilungen über Wochen hinweg kontaktiert werden, um konkrete Details zu Teilleistungen zu erfahren.

5.9 Fazit

Aus der Zusammenarbeit zwischen der DBIS, dem Studiendekanat der Medizin und dem kiz können vielerlei Erkenntnisse für künftige Projekte gewonnen werden. Auch bieten diese Erfahrungen einen Einblick in die berufliche Praxis. Wie in Abschnitt 5.8 beschrieben, können während eines Projektes, beziehungsweise einer Arbeit, immer wieder Probleme auftauchen, mit denen zuvor nicht gerechnet wurde. Auch müssen Anforderungen bis ins kleinste Detail besprochen und definiert werden, um unnötigen Programmieraufwand zu verhindern. Fachfremden Mitarbeitern müssen Problemstellen in der Umsetzung der Anforderungen einfach verständlich gemacht und zusammen ein Konzept erarbeitet werden, das beide Seiten zufriedenstellt. Leider lassen sich in Hinblick auf Urlaubstage, Tagungen und sonstige terminliche Einschränkungen, lange Antwortzeiten auf Mails oder Präsentationsterminen nie ausschließen. Die angesprochenen Schwierigkeiten sorgten auch dafür, dass bei der Umsetzung der Anforderungen Abstriche gemacht werden mussten, um im Laufe dieser Arbeit so viele Anforderungen wie möglich umzusetzen. Bevor in Kapitel 8 ein Abgleich der Anforderungen stattfindet, wird zunächst im folgenden Kapitel die Gesamtarchitektur und im darauffolgenden Kapitel einzelne Implementierungsdetails zu umgesetzen Anforderungen vorgestellt.

6 Gesamtarchitektur

Dieses Kapitel soll einen Einblick in die Gesamtarchitektur der in dieser Arbeit verwendeten Systeme geben. Zunächst wird dazu in Abbildung 6.1 ein Überblick über das Zusammenspiel aller beteiligten Systeme gegeben und in den folgenden Abschnitten das Zusammenspiel aus der jeweiligen Sichtweise jedes dieser Elemente erläutert. Ein tieferer Einblick in das System erfolgt erst im nächsten Kapitel.

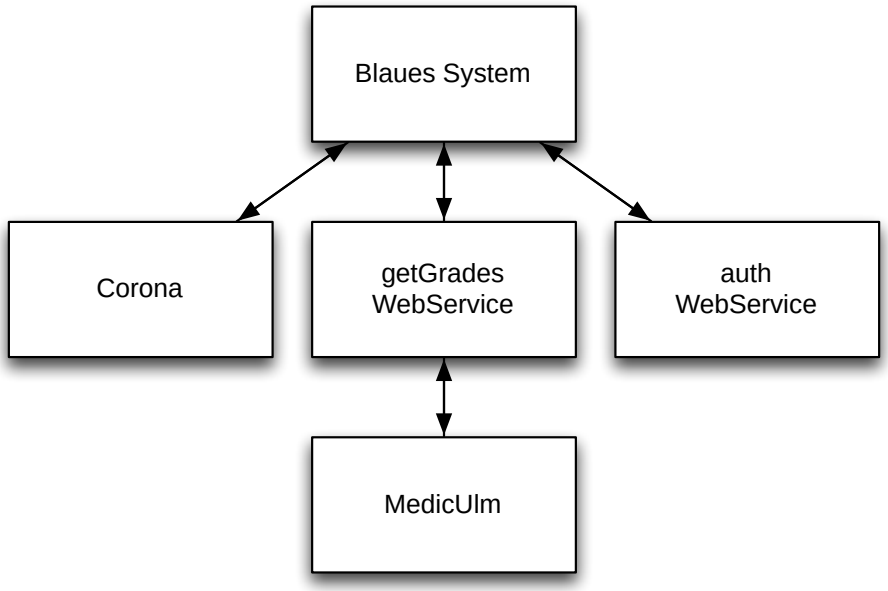


Abbildung 6.1: Ein Überblick über die Gesamtarchitektur

6.1 Datenbank Blaues System

Für die Umsetzung der Anforderungen konnten zum Teil die bereits vorhandenen Datenbanktabellen verwendet werden. Für einen Großteil der Funktionalität mussten zusätzliche Tabellen erstellt und vorhandene erweitert werden. Um einen Einblick in die Struktur des Systems zu erhalten, wird im nächsten Abschnitt ein zen-

traler Teil der Datenbankstruktur und im darauffolgenden Abschnitt die in dieser Arbeit eingebrachten Erweiterungen der Datenbank vorgestellt.

6.1.1 Student als zentrales Element

In Abbildung 6.2 wurden alle mit der Tabelle Student direkt zusammenhängenden Tabellen dargestellt. Das bedeutet, dass alle Tabellen entweder den Primärschlüssel oder eines seiner Attribute referenzieren. Darunter fallen unter anderem Log-Tabellen, welche die Eingaben des Benutzers protokollieren. Die Tabelle *LogStudent* hält beispielsweise Noten-Eintragungen für Studenten fest. Weitere Tabellen sind Zwischenrelationen wie *VeranstaltungStudent*, die die Note zu einer abgelegten Veranstaltung speichert. Die restlichen Tabellen beziehen sich auf Zuweisungen bezüglich des Studienverlaufs. Hier wird als Beispiel in der Tabelle *Student* im Attribut *StudienordnungID_VK* eine Studienordnung aus der Tabelle *Studienordnung* zugeordnet. Eine komplette Übersicht über alle Tabellen findet sich im Anhang in Kapitel 9.2 dieser Arbeit.

6.1.2 Erweiterungen in dieser Arbeit

Neben den bereits vorhandenen Tabellen war es nötig, die Datenbankstruktur sowohl um weitere Tabellen zu erweitern als auch bestehende Tabellen um Attribute zu erweitern. Die Erweiterungen sind in Abbildung 6.3 dargestellt. Hierzu werden die Anforderungen nochmals aufgeführt und die dafür neu erstellten oder erweiterten Tabellen beschrieben.

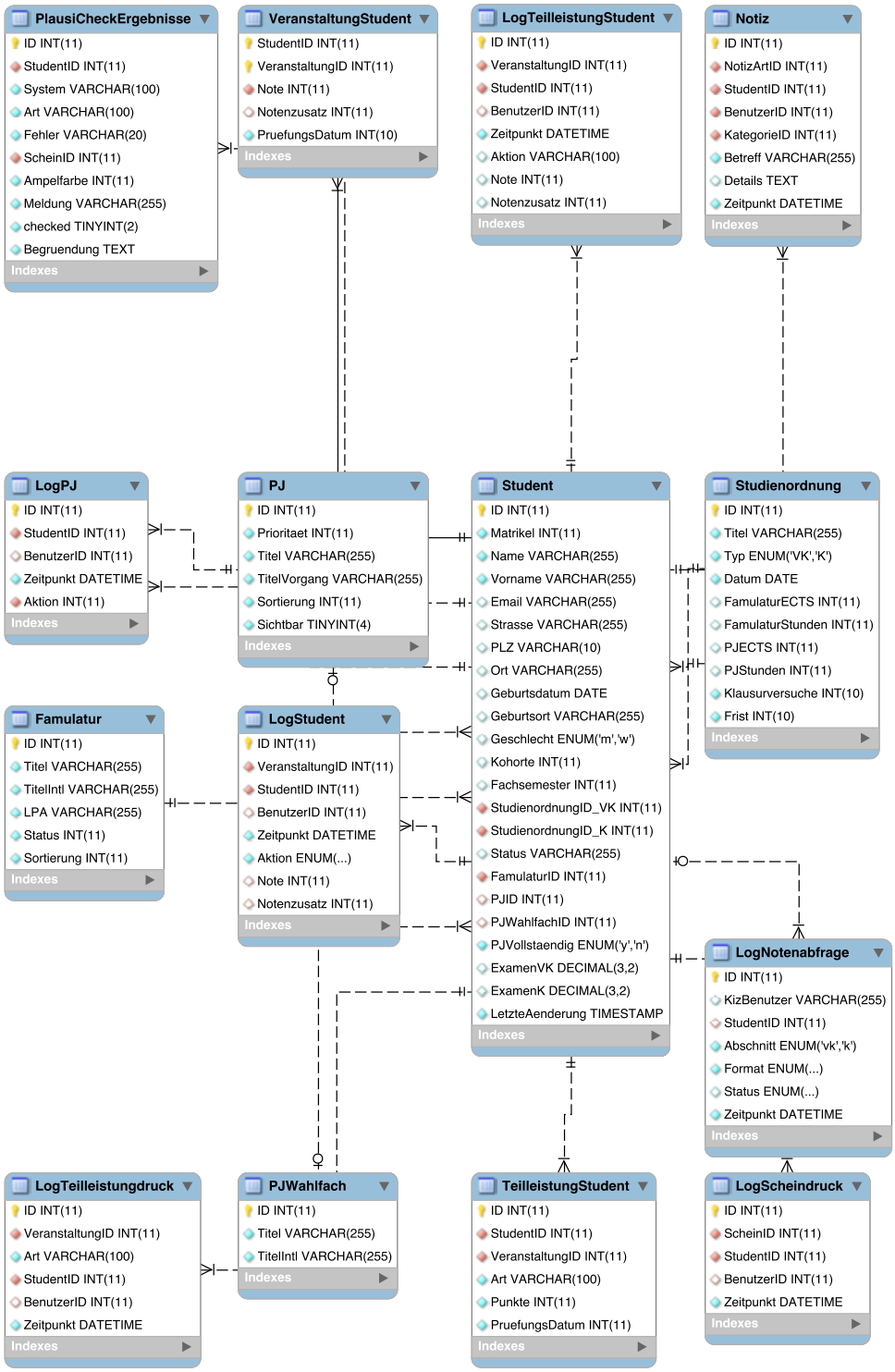


Abbildung 6.2: Datenbank-Schema bezogen auf Tabelle Student

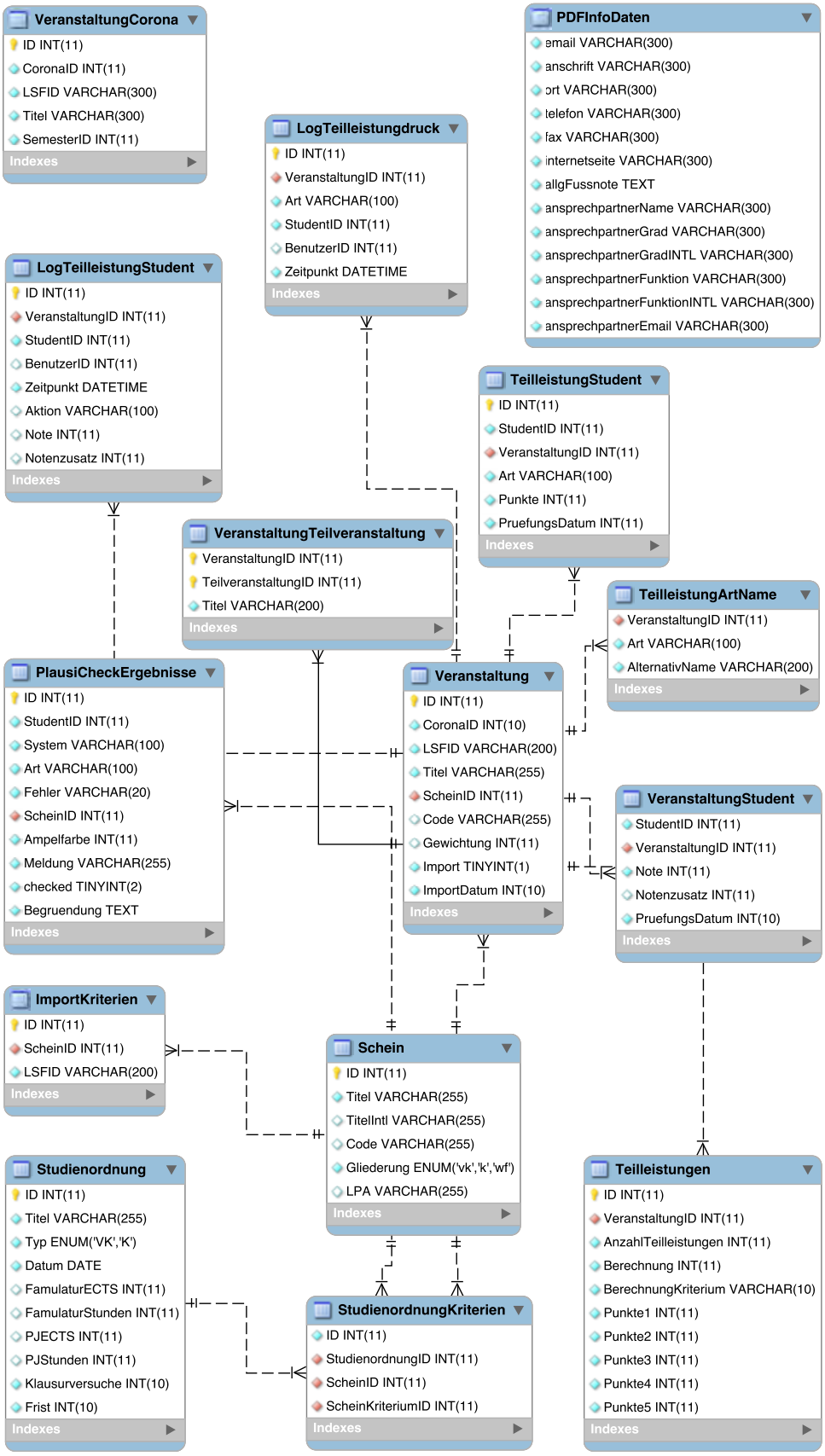


Abbildung 6.3: Erweitertes Datenbank-Schema

- Teilleistungen
  - Hierfür wird zunächst die Tabelle *Teilleistungen* für die Speicherung der angelegten Teilleistungen verwendet. Da jede erstellte Teilleistung mehrere ihr zugeordnete Teilleistungen besitzen kann, wird in *TeilleistungArtName* der dafür angelegte Name und die Art der Teilleistung eingetragen. *TeilleistungStudent* hält die eingetragenen Noten oder Punkte der Teilleistung bereit. Für eingetragene Teilleistungen wird eine Log-Meldung in *LogTeilleistungStudent* angelegt. Bereits gedruckte Nachweise für Teilleistungen werden in der Tabelle *LogTeilleistungdruck* erfasst.
- Studienordnung
  - Die bereits vorhandene Tabelle *Studienordnung* wurde um die Attribute Klausurversuche und Frist erweitert, um die für jede Studienordnung definierten Fristen für Klausuren und Wiederholungszeiträume zu erfassen. Die Voraussetzungskriterien, sofern für einen Schein vorhanden, sind in der Tabelle *StudienordnungKriterien* eingetragen.
- Plausibilitäts-Checks
  - Bevor mit den Checks begonnen werden kann, müssen zunächst Zuordnungen seitens Scheinen und Voraussetzungen definiert werden. Abbildung 6.4 zeigt den Ablauf der Zuordnungen, bevor die eigentlichen Checks stattfinden. Die bereits beschriebenen Tabellen *Studienordnung* und *StudienordnungKriterien* werden für die Checks ausgelesen. Wird ein Fehler gefunden, wird dieser in *PlausiCheckErgebnisse* eingetragen.

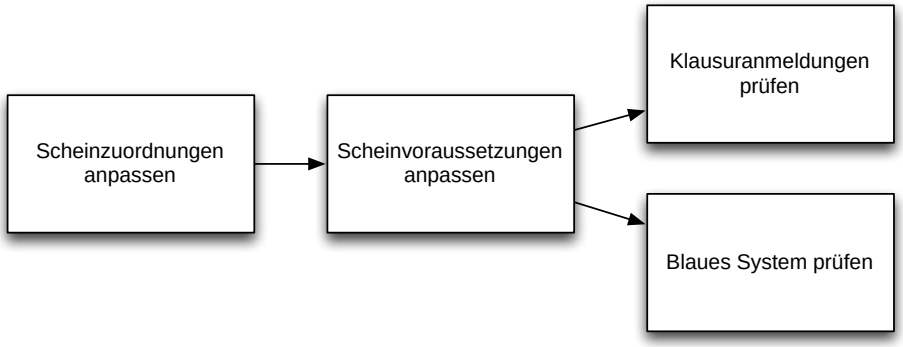


Abbildung 6.4: Ablaufdiagramm der Plausibilitäts-Checks

- Studentenansicht
  - Die Kurzansicht der Fehler aus den Plausibilitäts-Checks in der Studentansicht werden für jeden Studenten anhand der bereits erwähnten Tabelle *PlausiCheckErgebnisse* überprüft.
- PDFs
  - Die editierbaren Infos, die bei der Erzeugung von Nachweisen und Bescheinigungen automatisch eingefügt werden, werden in der Tabelle *PDFInfoDaten* gespeichert.
- Corona
  - Aus Corona zu importierende Veranstaltungen werden in der Tabelle *VeranstaltungCorona* gespeichert. Sofern für eine Veranstaltung Teilveranstaltungen existieren, werden diese in *VeranstaltungTeilveranstaltung* erfasst. Die Tabelle *Veranstaltung* wurde zusätzlich um das Attribut LSFID erweitert, das zum Abgleich der Import-Kriterien verwendet wird.

6.2 Blaues System und Corona

Das Blaue System ist das zentrale Element des Gesamtsystems. Das Sequenzdiagramm in Abbildung 6.5 zeigt die Schnittstelle zwischen Blauem System und Corona. Aus Corona wird mittels eines GET-Requests eine Veranstaltungsliste angefordert. Dieser GET-Request ist beispielhaft in Listing 6.1 aufgeführt und wird in Abschnitt 7.1.3 im folgenden Kapitel näher erklärt. Diese XML-Veranstaltungsliste wertet das Blaue System anschließend aus und speichert die Veranstaltungen und deren Teilveranstaltungen in der Datenbank des Blauen Systems. Diese XML-Liste ist in Listing 6.2 zu sehen. Corona kann über diese Teilveranstaltungen ebenfalls via GET abgefragt werden und gibt eine XML-Liste angemeldeter Studenten zurück. Um unbefugten Zugriff zu verhindern, ist der Zugriff auf die Schnittstelle auf zuvor definierte IPs beschränkt.

Listing 6.1: GET-Request zum Import von Veranstaltungen

```
1 https://campusonline.uni-ulm.de/CoronaNG/service.html?action
    =baum&semester=20122&uid=cgrab
```

Listing 6.2: XML-Veranstaltungsliste aus Corona

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <answer>
```

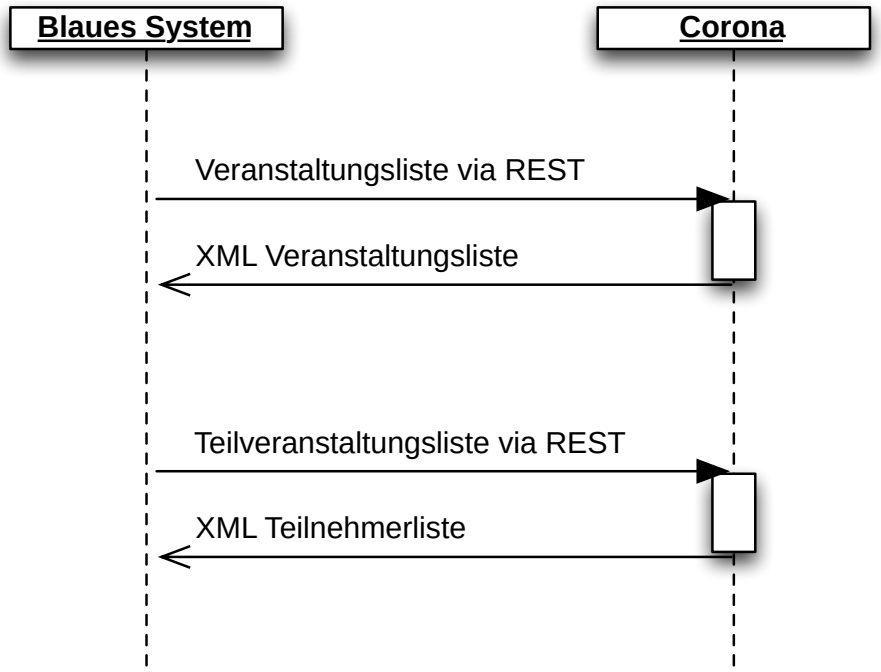


Abbildung 6.5: Zusammenspiel Blaues System - Corona

```
3     <tree>
4         <ueberschrift id="9947" name="Vorlesungsverzeichnis"
              >
5         <ueberschrift id="10044" name="Fakultaet␣fuer␣
              Mathematik␣und␣Wirtschaftswissenschaften">
6         <ueberschrift id="9912" name="Mathematik/
              Wirtschaftsmathematik">
7         <ueberschrift id="10392" name="Bachelor">
8             <lsfveranstaltung id="49532" name="MED500:␣
                  Clinical␣trials">
9                 <coronaveranstaltung id="2155658" name="
                      Clinical␣trials">
10                    <teilveranstaltung id="2155661" name="
                          Clinical␣trials">
11                    </teilveranstaltung>
12                </coronaveranstaltung>
13            </lsfveranstaltung>
14        </ueberschrift>
15        </ueberschrift>
```

```
16          </ueberschrift>
17          </ueberschrift>
18      </tree>
19  </answer>
```

6.3 MedicUlm, WebServices und Blaues System

Der Ablauf der Notenabfrage innerhalb der *MedicUlm* Android App ist in Abbildung 6.6 dargestellt.

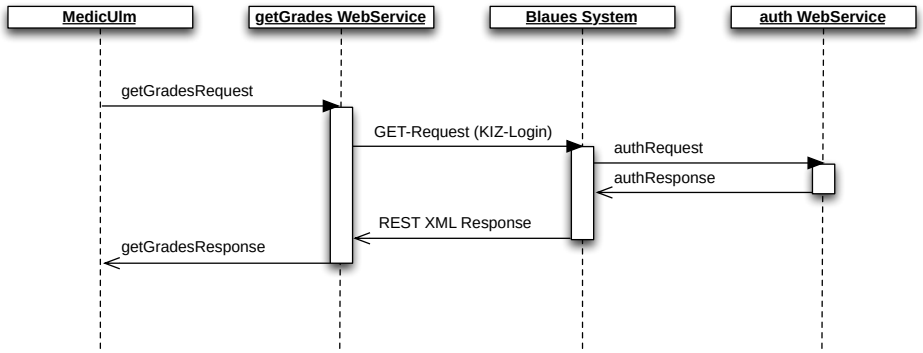


Abbildung 6.6: Zusammenspiel MedicUlm - WebServices - Blaues System

Von der Android App aus werden die kiz-Login Daten des Studenten in eine SOAP Message verpackt und an den *getGrades WebService* übergeben. Diese SOAP Nachricht ist in Listing 6.3 abgebildet.

Listing 6.3: getGradesRequest SOAP-Nachricht

```
1  <v:Envelope xmlns:i="http://www.w3.org/2001/XMLSchema-
       instance" xmlns:d="http://www.w3.org/2001/XMLSchema"
       xmlns:c="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:v="http://schemas.xmlsoap.org/soap/envelope/"><
       v:Header/>
2      <v:Body>
3          <n0:getGrades id="o0" c:root="1" xmlns:n0="http://
               ttdev.com/ls">
4              <user i:type="d:string">s_mmustermann</user>
5              <pwd i:type="d:string">dDNtcHC4</pwd>
6          </n0:getGrades>
7      </v:Body>
```

```
8 </v:Envelope>
```

Dieser WebService ruft das Blaue System mittels eines GET-Requests auf und gibt die Login-Daten dabei an das System weiter. Das Blaue System wiederum nutzt den in der DBIS-Abteilung bereits implementierten *auth WebService* um die Login-Daten zu überprüfen. Abhängig davon wird entweder eine XML-Notenliste oder ein Fehler an den getGrades WebService zurückgegeben, der diese Antwort wiederum an die Android App weiterleitet. Werden die Login-Daten erfolgreich authentifiziert, erhält die MedicUlm App eine SOAP-Response Nachricht, wie in Listing 6.4 zu sehen ist. Hier wurden Details zu den Scheinen ausgespart. Eine vollständige Auflistung der Notenliste findet sich im folgenden Kapitel im Abschnitt 7.3.2.

Listing 6.4: getGradesRequest SOAP-Nachricht

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
       soap/envelope/">
3      <soapenv:Body>
4          <ns1:getGradesResponse xmlns:ns1="http://ttdev.com/
               ss">
5              <?xml version="1.0" encoding="UTF-8"?>
6                  <root>
7                      <vorklinik>[...]</vorklinik>
8                      <klinik>[...]</klinik>
9                  </root>
10         </ns1:getGradesResponse>
11     </soapenv:Body>
12 </soapenv:Envelope>
```

6.4 Fazit

Nachdem in diesem Kapitel die Gesamtarchitektur hinsichtlich des Blauen Systems, der WebServices und der MedicUlm App angesprochen wurde, werden im folgenden Kapitel Implementierungsdetails aus den genannten Systemen beschrieben. Hier wird ein näherer Einblick in konkrete Details der Programmierung gegeben.

7 Implementierung

Nachdem in den vorherigen Kapiteln die Anforderungen und die Gesamtarchitektur beschrieben wurden, wird nun auf ausgewählte Implementierungsdetails eingegangen. Dazu werden die folgenden Abschnitte in die beteiligten Systeme aufgeteilt und einzelne Funktionen und Details zur Implementierung vorgestellt.

7.1 Blaues System

In diesem Abschnitt werden die wichtigsten Punkte der Programmierung des Blauen Systems dargestellt. Hierbei werden die einzelnen Anforderungen aufgezählt und zu jeder Anforderung die wichtigsten Implementierungsdetails aufgezeigt.

7.1.1 Teilleistungen

Die Verwaltung von Teilleistungen ist in die Menüpunkte *Teilleistungen verwalten* und *Teilleistungen importieren* untergliedert. In der Verwaltung können neue Teilleistungen für Veranstaltungen erstellt und vorhandene bearbeitet werden. Diese Werte werden in der Tabelle *Teilleistungen* gespeichert. Die Attribute hierfür sind *ID*, *VeranstaltungID*, *AnzahlTeilleistungen*, *Berechnung*, *BerechnungKriterium* und die *Punkte1* bis *Punkte5*. Die *ID* ist der Primärschlüssel der Relation. Im Attribut *VeranstaltungID* wird die zugeordnete Veranstaltung gespeichert, für die diese Teilleistungen definiert werden. Die *AnzahlTeilleistungen* definiert, wieviele Teilleistungen für die Veranstaltung existieren. Anhand dieses Wertes kann überprüft werden, ob alle Teilleistungen einer Veranstaltung vorliegen und daraus die Endnote berechnet werden. Die Art dieser Berechnung wird im Attribut *Berechnung* festgehalten. Bei der Erstellung als auch Bearbeitung von Teilleistungen kann angegeben werden, wie die Endnoten berechnet werden. Entweder als arithmetisches Mittel, kumuliert über einen bestimmten Gesamtwert oder über eine optionale Gewichtung. Diese Gewichtung legt die Grenzen dafür fest, ab welcher Punktzahl eine bestimmte Note vergeben wird. Beispielsweise erhält ein Student mit einer Gesamtpunktzahl größer gleich 90% der Maximalpunktzahl die Note *sehr gut*. Bei

einer Punktzahl unter 60% gilt die Veranstaltung als *nicht bestanden*. Diese Abstufungen werden in den Attributen *Punkte1* bis *Punkte5* festgehalten.
Im Menüpunkt *Teilleistungen importieren*, der dem vorhandenen Notenimport nachempfunden ist, können Teilleistungen importiert werden. Hierfür wird eine Auswahlliste der definierten Teilleistungen des ausgewählten Semesters angezeigt. Weist ein Student alle geforderten Teilleistungen einer Veranstaltung auf, wird beim Import die Endnote berechnet. Anhand der in der Tabelle *Teilleistungen* definierten Berechnungsmethode kann diese Endnote berechnet werden.
Die Berechnung der Endnoten findet sowohl in der Studentenansicht als auch beim Import der Teilleistungen statt. Hierzu kann in der Studentenansicht eine oder mehrere Teilleistungen bearbeitet werden, sodass die Endnote neu berechnet werden muss. Ebenfalls kann beim Import der Fall auftreten, dass Teilleistungen aktualisiert werden oder die geforderte Anzahl der Teilleistungen einer Veranstaltung erreicht wird. Die Berechnung der Endnote findet in der Datei *teilleistungen.inc* statt. Die Funktion *berechneGesamtnote()* überprüft anhand der übergebenen Veranstaltung, welche Berechnungsmethode verwendet werden soll. Je nach Berechnungsart werden die Funktionen *arithmetischesMittel()*, *kumuliertStudienordnung()* oder *kumuliertGewichtung()* aufgerufen. Beispielhaft wird nun die Berechnung der Gewichtung näher beschrieben.

Berechnung der gewichteten Endnote

Listing 7.1 zeigt die Selektion der Gesamtsumme und der Anzahl der vorhandenen Teilleistungen eines Studenten in der übergebenen Veranstaltung. In Zeile 3 wird die Art der Teilleistung eingeschränkt, damit eine mögliche Nachklausur nicht beachtet wird. Wurde eine Nachklausur eingetragen, wird die Berechnung der Endnote nicht aufgerufen. Dieses Abfrageergebnis wird nun an die Funktion *kumuliertGewichtung()* zusammen mit den Details der Berechnung übergeben. Diese enthalten alle Attribute der Tabelle *Teilleistungen*.

Listing 7.1: Selektion der Gesamtsumme und der Anzahl aller Teilleistungen eines Studenten

```
1 SELECT SUM(Punkte) AS Pruefungspunkte, COUNT(ID)
    AnzahlPruefungen
2 FROM TeilleistungStudent
3 WHERE StudentID = '.$studentID.'  AND VeranstaltungID = '.
    $aktVeranstaltung.' AND Art LIKE 'Teilleistung%'
4 HAVING COUNT(ID)
```

In der Funktion selbst wird zunächst in Zeile 2 die für die Teilleistungen definierte Maximalanzahl der erreichbaren Punkte extrahiert. Darauffolgend wird die Anzahl der definierten Teilleistungen der Veranstaltung und die Gesamtnote aller eingetragenen Prüfungspunkte gespeichert. Zuletzt wird in Zeile 5 die Anzahl der abgelegten Teilleistungen in der Variable *$countTeilleistungen* gespeichert. Diese wird anschließend in Zeile 7 mit der Anzahl der definierten Teilleistungen verglichen. Stimmen die Werte der Variablen überein, erfolgt die eigentliche Berechnung der Endnote. Dazu wird zunächst in Zeile 8 aus der Gesamt- und Maximalpunktzahl der abgelegten Teilleistungen ein Prozentwert gebildet. Dieser Prozentwert wird mit den zuvor für die Teilleistungen definierten Gewichtungen verglichen. Konkret wird in if-Bedingungen überprüft, ob die Prozentzahl einen definierten Schwellenwert erreicht. Ist dies der Fall, wird der entsprechende Notenwert als Endnote gespeichert und in Zeile 22 zurückgegeben.

Listing 7.2: Berechnung der Endnote anhand der definierten Gewichtung

```
1  function kumuliertGewichtung($teilleistungen,
       $berechnungsDetails){
2      $gesamtPunktzahl = $berechnungsDetails[0]['
           BerechnungKriterium'];
3      $anzahlTeilleistungen = $berechnungsDetails[0]['
           AnzahlTeilleistungen'];
4      $gesamtNote = $teilleistungen[0]['Pruefungspunkte'];
5      $countTeilleistungen = $teilleistungen[0]['
           AnzahlPruefungen'];
6
7      if ($countTeilleistungen == $anzahlTeilleistungen){
8          $prozent = ($gesamtNote / $gesamtPunktzahl) * 100;
9
10         if ($prozent >= $berechnungsDetails[0]['Punkte1'])
11             $gesamtNote = 100;
12         else if ($prozent >= $berechnungsDetails[0]['Punkte2
               '])
13             $gesamtNote = 200;
14         else if ($prozent >= $berechnungsDetails[0]['Punkte3
               '])
15             $gesamtNote = 300;
16         else if ($prozent >= $berechnungsDetails[0]['Punkte4
               '])
17             $gesamtNote = 400;
```

```
18          else if ($prozent > $berechnungsDetails[0]['Punkte5']
                ])
19              $gesamtNote = 500;
20      }
21
22      return $gesamtNote;
23  }
```

7.1.2 Studienordnung

Die geforderten Funktionalitäten sind in zwei Menüpunkte ausgelagert. Zum einen *Prüfungsordnung verwalten*, zum anderen die Funktion *Scheinvoraussetzungen anpassen* als Teilbereich der Plausibilität-Checks. Die nachfolgend besprochenen Datenbankrelationen sind zur Übersicht in Abbildung 7.1 dargestellt.

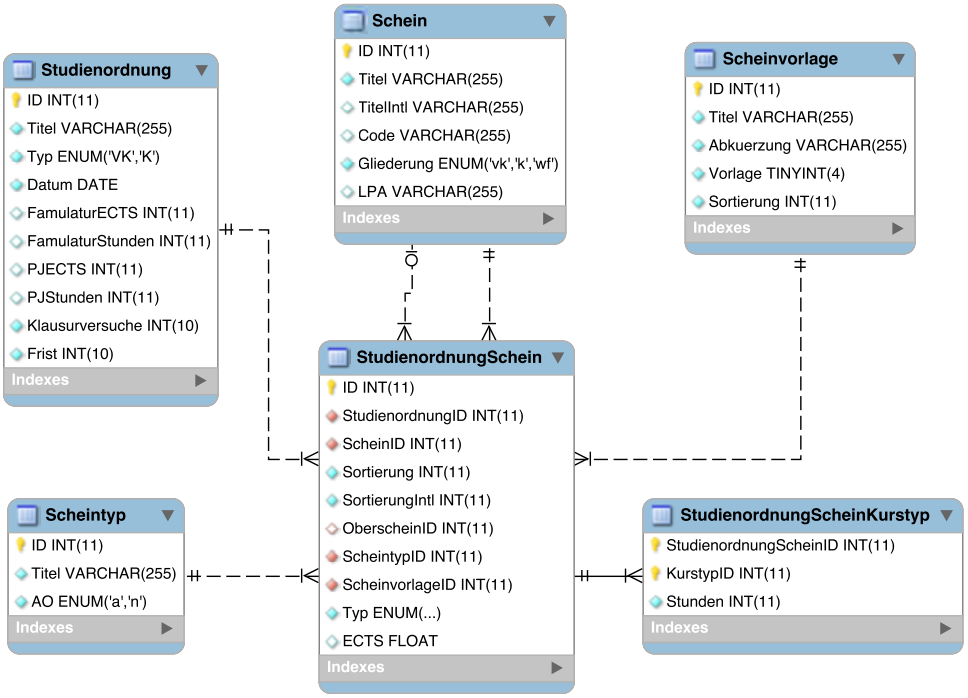


Abbildung 7.1: Die für die Studienordnung relevanten Relationen

Prüfungsordnung verwalten

Aus der Datenbank wird eine Liste der vorhandenen Studienordnungen aus der Tabelle *Studienordnung* an ein *Smarty Template* übergeben und in einer Auswahlliste angezeigt. Diese Studienordnungen können entweder bearbeitet oder kopiert

werden, um eine neue Studienordnung in die Datenbank einzufügen. Im Falle einer Bearbeitung werden anhand der ID der Studienordnung alle zugehörigen Scheine geladen. Zusätzlich werden alle restlichen Scheine angezeigt, die dem Typ der Studienordnung entsprechen. Dieser Typ wird unterschieden in Vorklinik und Klinik. Diese Scheine können dann der Studienordnung zugewiesen werden. Für diese neuen Zuordnungen kann anschließend die Sortierung, der Oberschein, Scheintypen, eine Scheinvorlage und ECTS Punkte mit angegeben werden.
Soll eine Studienordnung kopiert werden, werden nach der Angabe des Titels, der Vorklinik/Klinik und sowohl der Famulatur als auch den ECTS-Punkten und -Stunden des Praktischen Jahres, alle vorhandenen Zuordnungen der ausgewählten Studienordnung kopiert und als neue Studienordnung eingefügt.

Scheinvoraussetzungen anpassen

Da für die Plausibilitäts-Checks die Klausurversuche und -fristen von Bedeutung sind, wurde die Tabelle *Studienordnung* um diese Attribute erweitert. Diese Attribute können hier definiert und geändert werden.

7.1.3 Plausibilitäts-Checks

Eine Übersicht über den Ablauf vor den eigentlichen Checks findet sich in der Gesamtarchitektur in Kapitel 6, Abbildung 6.4. Nachfolgend werden die wichtigsten Implementierungsdetails dieser Funktionen beschrieben.

Scheinzuordnungen anpassen

Die Idee der Scheinzuordnung ist, aus Corona importierte Veranstaltungen mit Veranstaltungen des Blauen Systems abgleichen zu können. Daher muss für jede Veranstaltung, beziehungsweise für jeden vorhandenen Schein des Blauen Systems, eine manuelle Zuordnung stattfinden. Dafür wird aus der Datenbank die Tabelle *Schein* ausgelesen und die Scheine in einer Tabelle, sortiert nach Vorklinik und Klinik und deren Sortierung in der Studienordnung, angezeigt. Soll für einen Schein eine Zuordnung definiert werden, werden die bereits aus Corona importierten Veranstaltungen des ausgewählten Semesters angezeigt. Neue Zuordnungen werden in der Tabelle *ImportKriterien* gespeichert. Wurden bisher noch keine Veranstaltungen für das aktuelle Semester importiert oder sollen die vorhandenen Veranstaltungen aktualisiert werden, kann dies mit *Veranstaltungen aktualisieren* vorgenommen werden. Anschließend wird die Funktion *importVeranstaltungen()*

aufgerufen, die in der PHP Datei inkludiert ist. Die GET Anfrage zum eigentlichen Import der Veranstaltungen aus Corona ist in Listing 7.3 dargestellt. Corona liefert auf diese Anfrage hin einen Veranstaltungsbaum im XML-Format zurück. Diese Veranstaltungen entsprechen dem übergebenen Semester. Die Parameter werden in Abschnitt 7.2 genauer erklärt. Der String, den die *file_get_contents()* Funktion schließlich zurückgibt, wird nun als DOMDocument geparst und in Zeile 7 alle Tags inklusive deren Subtags in eine NodeList in der Variable *$lsfs* gespeichert. Die Tags werden anschließend in einer Schleife ausgelesen und die enthaltenen Daten wie *CoronaID*, *LSFID* und *Titel* in der Tabelle *VeranstaltungCorona* gespeichert. Sofern bereits Zuordnungen existieren, werden die Veranstaltungen des Blauen Systems mit den passenden Corona-Veranstaltungen abgeglichen und die *CoronaID* und *LSFID* in der Tabelle *Veranstaltung* aktualisiert. Zudem werden die in den aus Corona importierten Veranstaltungen enthaltenen Subelemente, hier Teilveranstaltungen genannt, ausgelesen. Diese werden in die Tabelle *VeranstaltungTeilveranstaltung* eingefügt. Anhand der Teilveranstaltungen können in den eigentlichen Plausibilitäts-Checks die angemeldeten Studenten aus Corona abgefragt werden.

Listing 7.3: Corona-Aufruf zum Import von Veranstaltungen

```
1 $requestURI = 'https://campusonline.uni-ulm.de/CoronaNG/
    service.html?action=baum&semester='.$semesterUri.'&uid=
    cgrab';
2 $ctx=stream_context_create(array('http'=>array('timeout' =>
    300)));
3 $str = file_get_contents($requestURI, false, $ctx);
4
5 $doc = new DOMDocument();
6 $doc->loadXML($str);
7 $lsfs = $doc->getElementsByTagName('lsfveranstaltung');
```

Scheinvoraussetzungen anpassen

Durch den Corona-Zugriff ist es möglich, Listen von für Veranstaltungen angemeldete Studenten aus denjenigen Veranstaltungen zu importieren, für die korrekte Scheinzuordnungen definiert wurden. Einige der Veranstaltungen, beziehungsweise der zugehörigen Scheine, setzen den Nachweis anderer Scheine voraus, um an der Prüfung teilnehmen zu dürfen. Daher können für jede Studienordnung Scheinvoraussetzungen definiert werden. Dazu wird in der Tabelle *StudienordungKrite-*

rien zu jeder *StudienordnungID* die *ScheinID* und die *ScheinKriteriumID* gespeichert. Letztere repräsentiert die ID eines Scheins, der für die Prüfung der Veranstaltung vorausgesetzt wird.

Prüfung des Blauen Systems

Nachdem die Scheinzuordnungen und Scheinvoraussetzungen festgelegt wurden, können die eigentlichen Checks durchgeführt werden. Zunächst kann der Datenbestand des Blauen Systems geprüft werden. Hier wird überprüft, ob es für die bereits eingetragenen Scheine oder Teilleistungen Verstöße gegen die in den Studienordnungen definierten Klausurversuche und -fristen gibt. Die Überprüfung der Scheine findet ausgehend von den Scheinen statt, die als *nicht ausreichend* eingetragen wurden. Bei Teilleistungen werden alle Teilleistungen berücksichtigt.
Die Überprüfung der Scheine und Teilleistungen wird voneinander unabhängig durchgeführt. Zunächst werden alle Studenten inklusive deren kritische Scheinveranstaltungen aus der Datenbank extrahiert. Das Kritierium hierfür ist eine nicht bestandene Prüfung. Listing 7.4 zeigt die zugehörige SQL-Anfrage. In Zeile 1 werden alle für die Plausibilitäts-Checks benötigten Werte aus den *gejointen* Tabellen selektiert. Die WHERE-Bedingung in Zeile 9 reduziert die Ergebnisliste auf die nicht bestandenen Prüfungen. Um die Ergebnismenge für die Checks entsprechend sortiert vorzubereiten, werden die selektierten Werte in Zeile 10 und 11 nach *StudentID* und *ScheinID* gruppiert und nach weiteren Attributen sortiert. In diesem Fall werden die *StudentIDs* absteigend sortiert. Anschließend wird zuerst nach dem *Jahr* und der *Art* des jeweiligen Semesters aufsteigend sortiert. Letzteres steht für die Angabe des Winter- oder Sommersemesters. Dies soll die Ergebnismenge daraufhin optimieren, dass die entsprechenden Veranstaltungen, für jeden Studenten chronologisch sortiert sind.

Listing 7.4: Alle Studenten, die eine Prüfung nicht bestanden haben

```
1 SELECT vst.StudentID, s.Vorname, s.Name, v.ScheinID, sch.
    Titel, vst.VeranstaltungID, sem.VorlesungAnfang, sem.
    VorlesungEnde, sem.Jahr, sem.Art, so.Klausurversuche, so
    .Frist
2 FROM VeranstaltungStudent AS vst
3 JOIN Student AS s ON vst.StudentID = s.ID
4 JOIN Veranstaltung AS v ON vst.VeranstaltungID = v.ID
5 JOIN VeranstaltungSemester AS vs ON v.ID = vs.
    VeranstaltungID
6 JOIN Semester AS sem ON vs.SemesterID = sem.ID
```

```
7  JOIN Schein AS sch ON v.ScheinID = sch.ID
8  JOIN Studienordnung AS so ON s.StudienordnungID_VK = so.ID
9  WHERE vst.Note = 500
10 GROUP BY vst.StudentID, v.ScheinID
11 ORDER BY vst.StudentID DESC, sem.Jahr ASC, sem.Art ASC, v.
       ScheinID ASC, vst.VeranstaltungID ASC
```

Diese Ergebnismenge wird nun an die eigentliche Methode *checkFristKlausurversuche()* für die Überprüfung der Wiederholungsfristen und Klausurversuche übergeben, die in der Datei *plausichecks.inc* ausgelagert ist. Diese speichert für die Fristüberprüfung zunächst das aktuelle Datum in der Variable *$aktSemester*. Anschließend wird das Array der Datenbank-Ergebnisse in einer *foreach* Schleife durchlaufen. In dieser Schleife werden nun alle nicht bestandenen Prüfungen durchlaufen. Um zu überprüfen, ob Überschreitungen für Fristen oder Klausurversuche vorliegen, wird anhand der *StudentID* und *ScheinID* geprüft, ob noch weitere Prüfungsleistungen für den Studenten vorliegen. Möglicherweise wurde eine spätere Prüfung desselben Scheins bestanden, sodass keine Überschreitung vorliegt. Die Datenbank-Anfrage ist in Listing 7.5 aufgeführt. Hier werden über JOINS alle Prüfungen eines Studenten selektiert. Die WHERE-Bedingung in Zeile 6 schränkt die Ergebnismenge in der Hinsicht ein, dass alle Prüfungen selektiert werden, die dieselbe *ScheinID* und *StudentID* aufweisen. Die *VeranstaltungID* muss hier allerdings unterschiedlich sein, da sonst die aktuell betrachtete, nicht bestandene Veranstaltung, nochmals ausgegeben wird.

Listing 7.5: Alle Prüfungen des Studenten desselben Scheins

```
1  SELECT vst.StudentID, v.ScheinID, vst.Note, vst.
       VeranstaltungID, sem.VorlesungAnfang, sem.VorlesungEnde,
        sem.Jahr, sem.Art
2  FROM VeranstaltungStudent AS vst
3  JOIN Veranstaltung AS v ON vst.VeranstaltungID = v.ID
4  JOIN VeranstaltungSemester AS vs ON v.ID = vs.
       VeranstaltungID
5  JOIN Semester AS sem ON vs.SemesterID = sem.ID
6  WHERE v.ScheinID = '.$e['ScheinID'].' AND vst.
       VeranstaltungID != '.$e['VeranstaltungID'].' AND vst.
       StudentID = '.$e['StudentID'].'
7  ORDER BY vst.StudentID ASC, v.ScheinID ASC, vst.
       VeranstaltungID ASC, sem.Jahr ASC, sem.Art ASC
```

Diese Ergebnismenge wird abermals in einer *foreach* Schleife durchlaufen. Die Schleife ist in Listing 7.6 aufgeführt. In Zeile 7 wird überprüft, ob sich die aktuelle Veranstaltung noch innerhalb der Wiederholungsfrist befindet und bestanden wurde. Wenn ja, wird die nicht bestandene Veranstaltung als bestanden angesehen. Wenn die Veranstaltung in Zeile 10 ebenfalls nicht bestanden wurde, wird die Zahl der Klausurversuche inkrementiert. Sollten bei der Fristüberprüfung Ungereimtheiten auftauchen, wird dies in Zeile 13 abgefangen und als Fehler gespeichert. In diesem Fall wurde eine Prüfung bereits zu einem früheren Datum als bestanden eingetragen. Hier muss dann seitens der Mitarbeiter geprüft werden, warum diese Prüfung zunächst als bestanden und in einem späteren Semester als nicht bestanden eingetragen wurde. Wurden alle Veranstaltungen durchlaufen, die demselben Schein zugeordnet sind, wird in Zeile 19 überprüft, ob die Anzahl der Klausurversuche überschritten wurde. Ist dies der Fall, wird dieser Fehler gespeichert. In diesem Listing nicht erfasst, findet anschließend noch eine Überprüfung statt, ob die Wiederholungsfrist überschritten wurde. Dies wird für den Fall geprüft, wenn die Variable *$bestanden* auf false gesetzt ist. Außerdem wird überprüft, ob sich die Wiederholungsfrist in einem kritischen Zeitraum befindet. Dahingehend wird der Fehler mit der Kennzeichnung *AMPEL_GELB* gespeichert. In diesem Fall wird in der Tabelle der Fehleranzeige die entsprechende Zeile gelb eingefärbt. Der Methode *speichereFehler()* werden in Zeile 21 verschiedene Parameter übergeben, die beim Speichern des Fehlers berücksichtigt werden. Zunächst wird erfasst, wo der Fehler auftrat - hier bei einem Schein. Anschließend wird das System, hier das Blaue System, übergeben. Im Falle einer Überschreitung der Klausurversuche wird als nächster Parameter die Fehlerart, hier *KV*, übergeben. Danach folgt die *ScheinID*, *StudentID*, eine Fehlernachricht, eine Ampelfarbe und der Parameter *checked*. Falls *checked* übergeben wird, werden bereits eingetragene Fehler aktualisiert. Hier kann der Fall auftreten, dass beispielsweise die Wiederholungsfrist überschritten wurde, beziehungsweise sich die Nachricht bezüglich der Monatsangabe ändert.

Listing 7.6: Überprüfung nach Klausurversuch- oder Fristüberschreitungen

```
1 foreach($verg as $v){
2     $note = $v['Note'];
3     $veranstaltungID = $v['VeranstaltungID'];
4     $datumAktPruefung = $v['VorlesungAnfang'];
5
6     $zeitspanneInMonaten = zeitspanneInMonaten(
          $datumErstePruefung, $datumAktPruefung);
7     if (($zeitspanneInMonaten < $frist) && ($note != 500))
```

```
8           $bestanden = true;
9
10      if ($note == 500)
11          $versuche ++;
12
13      if ($zeitspanneInMonaten < 0){
14          $message = 'Ungereimtheit bei Fristueberpruefung
                fuer Schein '.$scheinTitel.'. Ueberpruefte
                Veranstaltung mit "nicht ausreichend" war
                zeitlich nach bestandener Pruefung';
15          speichereFehler('Schein', $system, 'UG', $scheinID,
                $studentID, $message, AMPEL_FEHLER, $checked);
16      }
17  }
18
19  if ($versuche > $klausurversuche){
20      $message = 'Anzahl Klausurversuche wurde mit '.$versuche
            .' von '.$klausurversuche.' erlaubten ueberschritten
            .';
21      speichereFehler('Schein', $system, 'KV', $scheinID,
            $studentID, $message, AMPEL_ROT, $checked);
22  }
```

Eine ähnliche Vorgehensweise findet bei der Überprüfung der Teilleistungen statt. Hier werden zunächst alle Teilleistungen selektiert. Anschließend wird das aktuelle Datum gespeichert. In der SQL-Anfrage der Teilleistungen wird ebenfalls der Vorlesungsanfang der jeweiligen Teilleistung selektiert. Anhand des aktuellen Datums und des Vorlesungsanfangs der Teilleistungen wird nun geprüft, ob eine Verletzung der Wiederholungsfrist vorliegt. Diese Überprüfung findet nur dann statt, wenn bisher noch keine Endnote zur überprüften Veranstaltung eingetragen wurde. Wird eine Überschreitung festgestellt, wird auch hier der entsprechende Fehler in die Datenbank eingetragen.

Prüfung der Klausuranmeldungen

Nachfolgend wird die Überprüfung der Klausuranmeldungen aus Corona vorgestellt. Hierfür relevant sind alle Veranstaltungen, für die im Blauen System bereits eine *CoronaID* hinterlegt wurde. Diese ID wird beim Aktualisieren der Veranstaltungen aus Corona bei den entsprechenden Veranstaltungen gespeichert. Die SQL-

Anfrage, mit der alle relevanten Veranstaltungen selektiert werden, ist in Listing 7.7 dargestellt. Hierfür werden alle Veranstaltungen abgefragt, die im aktuell ausgewählten Semester stattfinden und für die eine *CoronaID* hinterlegt ist.

Listing 7.7: Alle relevanten Veranstaltungen mit CoronaID selektieren

```
1 SELECT v.CoronaID, v.Titel
2 FROM Veranstaltung AS v
3 JOIN VeranstaltungSemester AS vs ON v.ID = vs.
     VeranstaltungID
4 WHERE vs.SemesterID = '.$semester.' AND v.CoronaID > 0
```

Diese Veranstaltungen werden in einer *foreach* Schleife durchlaufen und zu jeder Veranstaltung die entsprechenden Teilveranstaltungen aus der Datenbank geladen. Die Teilveranstaltungen werden über die in der Veranstaltung hinterlegte *CoronaID* angesprochen. Diese SQL-Anfrage ist in Listing 7.8 aufgeführt.

Listing 7.8: Alle Teilveranstaltungen anhand der CoronaID selektieren

```
1 SELECT vt.TeilveranstaltungID
2 FROM VeranstaltungTeilveranstaltung AS vt
3 JOIN Veranstaltung AS v ON v.ID = vt.VeranstaltungID
4 WHERE v.CoronaID = '.$aktCorona;
```

Mithilfe dieser Teilveranstaltungen, genauer gesagt mithilfe der selektierten *TeilveranstaltungIDs*, können aus Corona die angemeldeten Teilnehmer der Veranstaltung mit einem GET-Request abgefragt werden. Eine Veranstaltung kann dabei mehrere Teilveranstaltungen besitzen, daher wird das Ergebnis aus obiger Datenbank-Anfrage ebenfalls in einer *foreach* Schleife durchlaufen. Listing 7.9 zeigt die Schleife und den darin ausgeführten GET-Request. In Zeile 2 ist die Anfrage-URL angegeben. Hier wird als Parameter *tvid* angegeben, der die aktuelle *TeilveranstaltungID* aus obiger SQL-Anfrage übergeben bekommt. Als Ergebnis wird aus Corona eine XML-Teilnehmerliste zurückgegeben. Diese wird in ein *DomDocument* geparst. In der XML Datei sind Details zu den Studenten hinterlegt. Anhand der ebenfalls enthaltenen Matrikelnummern können in Zeile 10 diese Matrikelnummern anhand ihres Tag-Namens extrahiert werden. Diese Nummern werden in Zeile 14 in ein Array gespeichert. Anhand dieser Matrikelnummern können die in Corona angemeldeten Studenten den Studenten aus dem Datensatz des Blauen Systems zugeordnet werden. In Zeile 16 wird die Anzahl der angemeldeten Studenten festgehalten.

Listing 7.9: Teilnehmerlisten aus Corona abfragen

```
1  foreach ($erg as $e){
2      $requestURI = 'https://campusonline.uni-ulm.de/CoronaNG/
           service.html?action=teilnehmer&tvid='.$e['
           TeilveranstaltungID'];
3
4      $ctx = stream_context_create(array('http'=>array('
           timeout' => 300)));
5      $str = file_get_contents($requestURI, false, $ctx);
6
7      $doc = new DOMDocument();
8      $doc->loadXML($str);
9
10     $matrikelnummern = $doc->getElementsByTagName('
           matrikelnummer');
11
12     foreach ($matrikelnummern as $m) {
13         $matrikel = $m->nodeValue;
14         $matrikelArray[] = $matrikel;
15     }
16     $matrikelCount += sizeof($matrikelArray);
17 }
```

Das Array mit den aus Corona importierten Matrikelnummern wird nun an die *checkVoraussetzungen()* Methode übergeben, die ebenfalls in der *plausichecks.inc* Datei zu finden ist. Hier werden zunächst anhand der ebenfalls übergebenen *CoronaID* Informationen zu der Veranstaltung aus der Datenbank extrahiert. Diese Informationen betreffen die *ScheinID*, den *Titel* und die *Gliederung*. Letztere gibt an, ob die Veranstaltung, beziehungsweise der ihr zugeordnete Schein, der Vorklinik oder Klinik zugeordnet ist.
In einer *foreach* Schleife wird nun das Array mit den importierten Matrikelnummern durchlaufen. Hier findet zunächst eine Überprüfung statt, welcher Studienordnung der jeweilige Student zugeordnet ist. Dies geschieht deshalb, weil für unterschiedliche Studienordnungen jeweils andere Voraussetzungen definiert sein können. Listing 7.10 zeigt die dafür verwendete SQL-Anfrage. In Zeile 1 werden *ScheinKriteriumID*, *Vorname*, *Name*, *ID* und *Titel* extrahiert. Die *ScheinKriteriumID* entspricht dabei einer *ScheinID*. Diese ID wurde zuvor für einen Schein als Voraussetzung in der Tabelle *StudienordnungKriterien* gespeichert. Dort sind alle Scheine und deren zugehörige Scheinvoraussetzungen hinterlegt. Diese Voraus-

setzungen werden immer für eine Studienordnung definiert, weshalb in Zeile 3 ein JOIN über die *StudienordnungID* stattfindet. In der WHERE-Bedingung wird die Ergebnismenge für die Selektion des Namens des Studenten und dessen ID anhand der Matrikelnummer begrenzt. Zusätzlich werden nur die Scheine der Tabelle *StudienordnungKritieren* betrachtet, die den Schein der aktuell betrachteten Veranstaltung betreffen.

Listing 7.10: Scheinkriterien, Scheintitel und studentische Daten selektieren

```
1 SELECT sk.ScheinKriteriumID, s.Vorname, s.Name, s.ID, sch.
    Titel
2 FROM Student AS s
3 JOIN StudienordnungKriterien AS sk ON s.StudienordnungID_'.
    strtoupper($studienordnung).' = sk.StudienordnungID
4 JOIN Schein AS sch ON sch.ID = sk.ScheinKriteriumID
5 WHERE s.Matrikel = '.$m.' AND sk.ScheinID = '.$schein
```

Das Ergebnis der Anfrage ist eine Ergebnismenge mit einem oder mehreren Scheinkriterien. Anhand dieser Kritierien kann nun festgestellt werden, ob der Student alle Voraussetzungen, sprich Kriterien, erfüllt, um an der angemeldeten Prüfung teilnehmen zu dürfen. Diese Voraussetzungen sind erfüllt, wenn er für alle Scheinkriterien eine bestandene Prüfungsleistung vorweisen kann. Dementsprechend werden diese Kriterien in einer *foreach* Schleife überprüft. Hier wird für das jeweilige Scheinkriterium und den aktuell betrachteten Studenten überprüft, ob eine entsprechende Prüfungsleistung vorliegt. Diese SQL-Anfrage ist in Listing 7.11 aufgeführt. In der WHERE-Bedingung in Zeile 5 wird zunächst die Ergebnismenge auf die *ScheinID* des Scheinkriteriums eingeschränkt. Weiter werden nur die Prüfungsleistungen betrachtet, die dem jeweiligen Studenten zugeordnet sind. Diese muss zudem eine *Note* aufweisen, die ungleich der Zahl 500 ist. Somit werden nicht bestandene Prüfungen ausgeschlossen.

Listing 7.11: Anhand des Scheinkriteriums bestandene Prüfungen selektieren

```
1 SELECT v.Titel, v.CoronaID
2 FROM Veranstaltung AS v
3 JOIN VeranstaltungStudent AS vs ON v.ID = vs.VeranstaltungID
4 JOIN Student AS s ON vs.StudentID = s.ID
5 WHERE v.ScheinID = '.$scheinkriterium.' AND s.Matrikel = '.
    $m.' AND vs.Note != 500
```

Als letzter Schritt wird überprüft, ob für diese Anfrage Ergebnisse vorliegen. Wenn keine Ergebnisse vorliegen, bedeutet dies, dass das Scheinkriterium nicht erfüllt

wurde. Daher wird dieser Fehler in die Datenbank eingetragen. Dafür wird abermals die *speichereFehler()* Methode aufgerufen. Wie dem Listing 7.12 zu entnehmen ist, ändern sich dementsprechend auch die übergebenen Parameter. Der Fehler trat in der Überprüfung der *Voraussetzung* auf. Das übergebene System ist in diesem Fall Corona. Die Fehlerart ist als *VS*, sprich Voraussetzung, angegeben.

Listing 7.12: Speichern eines nicht erfüllten Scheinkriteriums

```
1 speichereFehler('Voraussetzung', $system, 'VS', $schein,
    $studentID, $message, AMPEL_ROT, $checked);
```

Unabhängig davon wird für alle übergebenen Studenten eine Überprüfung auf Klausurversuche und -fristen durchgeführt. Diese Überprüfung findet anhand der aus Corona importierten Matrikelnummern statt. Dafür wird die bereits beschriebene Methode *checkFristKlausurversuche()* wiederverwendet.

7.1.4 Kohortenansicht

Alle Veranstaltungen eines ausgewählten Semesters werden in einer Auswahlliste angezeigt. Nach Auswahl einer Veranstaltung werden alle Kohorten zur Auswahl angezeigt, in denen für mindestens einen Studenten Ergebnisse in dieser Veranstaltung vorliegen. Wird eine Kohorte ausgewählt, wird eine Liste aller Studenten dieser Kohorte angezeigt. Diese wird anhand dreier Kriterien sortiert:

- Studenten, die an dieser Prüfung teilgenommen haben.
- Studenten, die bereits an einer Prüfung teilgenommen haben, die demselben Schein zugeordnet ist.
- Studenten, die bisher noch nie an einer Prüfung dieses Scheins teilgenommen haben.

Um die Komplexität aus den Datenbank-Abfragen zu reduzieren, wurde die Anfrage in zwei Teilanfragen unterteilt. Zunächst wird hierzu für die ausgewählte Veranstaltung und die ausgewählten Kohorten eine sortierte Liste aller Studenten der Kohorte erstellt. Diese Abfrage ist in Listing 7.13 aufgeführt. Durch den LEFT OUTER JOIN in Zeile 3 werden alle Studenten aus der Tabelle *Student* erfasst, unabhängig davon, ob für sie ein Ergebnis-Tupel in der Tabelle *VeranstaltungStudent*, beziehungsweise dessen JOIN über die Tabelle *Notentext* in Zeile 4, existiert. Durch die Definition von CASES in Zeile 6 kann die Ergebnismenge anhand verschiedener Kriterien nacheinander sortiert werden. Diese Kriterien sind zum einen

die Teilnahme an dieser Prüfung, das Nichtbestehen einer Prüfung (*NotenID* ist 500) und die Studenten, die an dieser Veranstaltung nicht teilgenommen haben. Diese sortierten Einträge werden dann jeweils noch nach den Nachnamen der Studenten sortiert.

Listing 7.13: Alle Studenten, die an einer Prüfung teilgenommen haben

```
1 SELECT s.Kohorte, s.ID, s.Vorname, s.Name, nt.Text
2 FROM Student AS s
3 LEFT OUTER JOIN VeranstaltungStudent AS vs ON s.ID = vs.
    StudentID AND vs.VeranstaltungID = '.$aktVeranstaltung.'
4 LEFT OUTER JOIN Notentext AS nt ON vs.Note = nt.ID
5 WHERE s.Kohorte IN ('.$selected_kohorten.')
6 ORDER BY s.Kohorte ASC, CASE WHEN nt.ID IS NULL THEN s.Name
    END ASC, CASE WHEN nt.ID = 500 THEN s.Name END ASC, CASE
     WHEN nt.ID IS NOT NULL THEN s.Name END ASC
```

Unter den Studenten der SQL-Anfrage in Listing 7.13 sind auch Studenten, die bereits an einer vorherigen Prüfung teilgenommen haben. Damit diese nicht fälschlicherweise als nicht teilgenommen deklariert werden, muss die Ergebnismenge mit einer zweiten Ergebnismenge abgeglichen werden. Die SQL-Anfrage für die zweite Ergebnismenge ist dem Listing 7.14 zu entnehmen. Hier werden alle Studenten selektiert, für die eine Prüfungsleistung für eine Veranstaltung desselben Scheins existiert. Die ID der Veranstaltung muss hier allerdings ungleich der aktuell betrachteten Veranstaltung sein, um doppelte Einträge auszuschließen.

Listing 7.14: Alle Studenten, die in einem anderen Semester an derselben Scheinprüfung teilgenommen haben

```
1 SELECT DISTINCT s.ID, s.Kohorte, s.Vorname, s.Name, nt.Text,
     sem.Jahr, sem.Art
2 FROM Student AS s
3 JOIN VeranstaltungStudent AS vs ON s.ID = vs.StudentID
4 JOIN Notentext AS nt ON vs.Note = nt.ID
5 JOIN Veranstaltung AS v ON vs.VeranstaltungID = v.ID
6 JOIN VeranstaltungSemester AS vsem ON vs.VeranstaltungID =
    vsem.VeranstaltungID
7 JOIN Semester AS sem ON vsem.SemesterID = sem.ID
8 WHERE s.Kohorte IN ('.$selected_kohorten.') AND vs.
    VeranstaltungID != '.$aktVeranstaltung.' AND v.ScheinID
    = '.$scheinID.'
9 ORDER BY nt.ID ASC
```

7.1.5 PDFs

Die Erweiterungen beim Druck von Nachweisen und Bescheinigungen bedürfen keiner allzu ausführlichen Beschreibung. Daher werden die einzelnen Anforderungen kurz angesprochen.

Druck von Teilleistungen

Da sich die Anforderungen bezüglich der Teilleistungen zum Ende dieser Arbeit erneut geändert hatten, wurde auf eine tiefergehende Erweiterung des vorhandenen Codes verzichtet. Vielmehr wurde der bereits vorhandene Code geklont und für die Teilleistungen angepasst. Im Vergleich zum Scheindruck muss hier der Titel und die Note, beziehungsweise die Punktzahl aus der Datenbank selektiert und in der PDF angezeigt werden. Der Druck der Teilleistungen findet über einen GET-Aufruf der *scheindruckTL.php* statt, die die benötigten Daten aus der Datenbank selektiert und mit der TCPDF Library eine PDF Datei erstellt.

Bearbeitung von Kontaktdaten und Hinweisen

Die ehemals statischen Daten in den Bescheinigungen und Nachweisen werden nun in der Datenbank vorgehalten und können jederzeit im Menüpunkt *Ansprechpartner* bearbeitet werden. Dafür werden in den Dateien *leistungsnachweis.inc*, *leistungsuebersicht.inc*, *leistungsuebersichtQuerformat.inc* und *tor.inc* die ehemals statischen Texte nun beim Erzeugen einer PDF aus der Tabelle *PDFInfoDaten* ausgelesen.

7.2 Corona

Die Implementierung der REST-Schnittstelle seitens Corona wurde nach der Definition der Anforderungen bezüglich Parameter und Rückgabewerte vom kiz in Person von Herrn Aschoff vorgenommen. Listing 7.15 zeigt den Import einer Veranstaltungsliste aus Corona. Der Parameter *semester* setzt sich aus der Jahreszahl und dem Semester zusammen. Das Sommersemester entspricht der Zahl 1, das Wintersemester der Zahl 2. Somit wäre der Wert des Parameters für das Wintersemester 2012 entsprechend *20122*. Der Veranstaltungsbaum, der anhand der Parameter aufgebaut wird, ist abhängig vom Parameter *uid*. Diese BenutzerID, hier Frau Grab, hat auf das gesamte Veranstaltungsverzeichnis einen eingeschränkten

Zugriff, sodass nur die für das Blaue System relevanten Veranstaltungen zurückgegeben werden.

Listing 7.15: GET-Request zum Import von Veranstaltungen

```
https://campusonline.uni-ulm.de/CoronaNG/service.html?action
    =baum&semester=20122&uid=cgrab
```

Als Antwort wird eine XML-Veranstaltungsliste zurückgegeben, die bereits in Kapitel 6 in Listing 6.2 gezeigt wurde. Diese enthält die für das Blaue System relevanten Veranstaltungen und deren optionale Teilveranstaltungen. Anhand dieser Teilveranstaltungen können die dafür angemeldeten Studenten ebenfalls mit einer GET-Anfrage abgefragt werden. Diese ist in Listing 7.16 für die Teilveranstaltungs-ID *2804462* dargestellt, die der Parameter *tvid* übergeben bekommt.

Listing 7.16: GET-Request zum Import von Veranstaltungen

```
https://campusonline.uni-ulm.de/CoronaNG/service.html?action
    =teilnehmer&tvid=2804462
```

Die Antwort darauf ist eine XML-Liste aller angemeldeten Studenten dieser Teilveranstaltung, die somit auch für die ihr übergeordneten Veranstaltung angemeldet sind. Diese List ist aufgeführt in Listing 7.17.

Listing 7.17: XML-Liste der angemeldeten Studenten einer Teilveranstaltung

```
<answer>
    <teilnehmer>
        <uid>s_mmustermann</uid>
        <mail>max.mustermann(at)uni-ulm.de</mail>
        <matrikelnummer>123456</matrikelnummer>
    </teilnehmer>
</answer>
```

7.3 MedicUlm

7.3.1 Erfassung der Login-Daten

Dem Benutzer der App werden in der *Grades* Activity Eingabefelder für den kiz-Benutzernamen und das Passwort bereitgestellt. Der Benutzername wird zunächst in den *SharedPreferences* [45] abgespeichert, damit dieser beim nächsten Aufruf aus dem App-Speicher geholt wird und nicht mehr erneut eingegeben werden

muss. Anschließend werden die Login-Daten an die *GradesDetail* Activity weitergegeben, in der diese Daten in einem separaten Thread an die *LDAPController* Klasse übergeben werden. In dieser wird ein SoapObject erzeugt, das aus der ksoap2 Library für Android eingebunden wird [46]. Ksoap2 ist eine SOAP Client Library für Android [47]. Listing 7.18 erzeugt in Zeile 1 das SoapObject, das in Zeile 8 und 16 den Benutzernamen und das Passwort übergeben bekommt. Dieses Objekt wird dann in einen Soap Envelope verpackt. Der *HttpTransportSE* Klasse wird die URL zur WSDL-Datei des WebServices übergeben und durch Übergabe des Methoden-Namens und des Soap Envelopes der getGrades-WebService angesprochen.

Listing 7.18: SOAP Aufruf in Android

```
1  SoapObject request = new SoapObject(NAMESPACE, method);
2
3  PropertyInfo pInfoUsername = new PropertyInfo();
4  pInfoUsername.setName("user");
5  pInfoUsername.setType(String.class);
6  pInfoUsername.setValue(username);
7
8  request.addProperty(pInfoUsername);
9
10 PropertyInfo pInfoPassword = new PropertyInfo();
11 pInfoPassword.setName("pwd");
12 pInfoPassword.setType(String.class);
13
14 password = Base64Coder.encodeString(password);
15 pInfoPassword.setValue(password);
16 request.addProperty(pInfoPassword);
17
18 SoapSerializationEnvelope soapEnvelope = new
       SoapSerializationEnvelope(
19     SoapEnvelope.VER11);
20 soapEnvelope.setOutputSoapObject(request);
21 HttpTransportSE tns = new HttpTransportSE(URL);
22 try {
23     tns.call(NAMESPACE + "/" + method, soapEnvelope);
24     result = soapEnvelope.bodyIn.toString();
25 } catch (Exeception e) {
26     e.printStackTrace();
```

```
27 }
```

7.3.2 Auslesen und Anzeige der Notenliste

Ist die Authentifizierung seitens des Blauen Systems erfolgreich, erhält die App eine Notenliste als Antwort zurück. Diese liegt im XML-Format vor und wird in der *XML-Parser* Util-Klasse ausgelesen. Die Struktur kann dem Listing 7.19 entnommen werden.

Listing 7.19: XML-Schema der Notenliste

```
1  <vorklinik>
2      <schein>
3          <id>1</id>
4          <titel>Praktikum der Physik fuer Mediziner</titel>
5          <note></note>
6          <datum></datum>
7      </schein>
8      <schein>
9          <id>2</id>
10         <titel>Praktikum der Chemie fuer Mediziner und
               Zahnmediziner</titel>
11         <note></note>
12         <datum></datum>
13     </schein>
14     <schein>
15         <id>3</id>
16         <titel>Praktikum der Biologie fuer Mediziner</titel>
17         <note>gut</note>
18         <datum>16.02.2013</datum>
19     </schein>
20 </vorklinik>
21 <klinik>
22     [...]
23 </klinik>
```

Diese XML-Liste wird in der *XMLParser* Klasse ausgelesen und anschließend in eine ListView [48] [49] geladen. Diese ListView Komponente zeigt die Noten in einer scrollbaren Liste an. Wird die App in der Notenansicht pausiert oder gestoppt,

wird beim Resume der App wieder der Login angezeigt, um fremden Zugriff auf die Notenliste zu verhindern.

7.4 getGrades WebService

Für den WebService wird zunächst eine WSDL Datei definiert, die in Listing 7.20 aufgeführt ist. In den *types* werden die Datentypen der ausgehenden und eingehenden SOAP-Nachrichten definiert. Der *getGradesRequest* in Zeile 5 ist vom Typ *complexType*, da er zwei Strings für den Benutzernamen und das Passwort beinhaltet. Die *getGradesResponse* in Zeile 13 ist vom Typ String und gibt die Notenliste des Blauen Systems in XML-Form zurück. Die *getGradesRequest* und *getGradesResponse* Elemente werden in Zeile 18 und 21 als *Message* definiert. Diese Messages werden nun als eingehende und ausgehende Nachricht für die Operation *getGrades* als Schnittstelle definiert. Diese Operation ist vom Typ *Request-Response*. Anschließend wird die Operation im Binding angegeben, in dem das Protokoll und das Datenformat definiert wird. Der Service *GradesService*, der in Zeile 42 einen Port für das eben beschriebene Binding aus Zeile 29 bindet, verweist auf die Adresse unter der der Port erreichbar ist.

Listing 7.20: XML Schema der Notenliste

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/
       wsdl/soap/" xmlns:tns="http://ttdev.com/ss" xmlns:wsdl="
       http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.
       w3.org/2001/XMLSchema" name="SimpleService"
       targetNamespace="http://ttdev.com/ss">
3    <wsdl:types>
4      <xsd:schema targetNamespace="http://ttdev.com/ss">
5        <xsd:element name="getGradesRequest">
6          <xsd:complexType>
7            <xsd:sequence>
8              <xsd:element name="user" type="xsd:string"/>
9              <xsd:element name="pwd" type="xsd:string"></
                   xsd:element>
10           </xsd:sequence>
11         </xsd:complexType>
12       </xsd:element>
```

```
13      <xsd:element name="getGradesResponse" type="xsd:string
            ">
14      </xsd:element>
15    </xsd:schema>
16  </wsdl:types>
17  <wsdl:message name="getGradesRequest">
18    <wsdl:part element="tns:getGradesRequest" name="
          parameters"/>
19  </wsdl:message>
20  <wsdl:message name="getGradesResponse">
21    <wsdl:part element="tns:getGradesResponse" name="
          parameters"/>
22  </wsdl:message>
23  <wsdl:portType name="GradesService">
24    <wsdl:operation name="getGrades">
25      <wsdl:input message="tns:getGradesRequest"/>
26      <wsdl:output message="tns:getGradesResponse"/>
27    </wsdl:operation>
28  </wsdl:portType>
29  <wsdl:binding name="GradesServiceSOAP" type="
        tns:GradesService">
30    <soap:binding style="document" transport="http://schemas
          .xmlsoap.org/soap/http"/>
31    <wsdl:operation name="getGrades">
32      <soap:operation soapAction="http://ttdev.com/ss/
            NewOperation"/>
33      <wsdl:input>
34        <soap:body use="literal"/>
35      </wsdl:input>
36      <wsdl:output>
37        <soap:body use="literal"/>
38      </wsdl:output>
39    </wsdl:operation>
40  </wsdl:binding>
41  <wsdl:service name="GradesService">
42    <wsdl:port binding="tns:GradesServiceSOAP" name="p1">
43      <soap:address location="http://134.60.71.103:8080/
            axis2/services/GradesService/"/>
```

```
44      </wsdl:port>
45    </wsdl:service>
46  </wsdl:definitions>
```

Anhand dieser WSDL Datei kann sowohl der *Client Stub* als auch der *Server Skeleton* definiert werden. Eine Client-Implementierung erfolgt allerdings, wie bereits in Abschnitt 7.3 beschrieben, auf Android-Basis. Der *Server Skeleton* wird mithilfe des Java-basierten Apache Maven Build-Management-Tools erzeugt, für den ein Eclipse-Plugin existiert [50]. In der *GradesService* Skeleton Klasse muss die *getGrades* Methode mit Funktionalität erweitert werden. Im *getGradesRequest*, den die Methode übergeben bekommt, werden dazu die Login-Daten ausgelesen und an das Blaue System weitergegeben. Die XML-Liste, die als Antwort des Aufrufs zurückgegeben wird, wird dann in die *getGradesResponse* Nachricht verpackt. Dieser WebService wird auf dem Axis2 Server der DBIS angeboten. Axis2 ist die nächste Generation der Apache WebService Middleware, die vor allem die Leichtgewichtigkeit als Fokus hat. Sie ist eine SOAP-Engine, mit der WebServices als auch Client-Anwendungen kreiert werden können [51]. WebServices werden in den *services* Ordner deployed.

7.5 Problemstellen im Blauen System

Das Blaue System weist an einigen Stellen Stolperfallen auf, die bei der Einarbeitung in das System, beziehungsweise bei der Erweiterung hinderlich sind. Daher werden nun einige Funktionen und deren Zusammenspiel zum einfacheren Verständnis im Detail erläutert.

7.5.1 PDF-Erzeugung von Nachweisen und Bescheinigungen

In der Studentenansicht können je nach Art des Nachweises jeweils programmatisch PDFs generiert werden. Für diese Erzeugung wird zwar aus der Studentenansicht nur eine PHP Datei mittels GET-Parametern aufgerufen, diese bedient sich allerdings verschiedener .inc Dateien, die wiederum weitere Dateien inkludieren, als auch Funktionen kapseln. Dies führt dazu, dass der Überblick schnell verloren geht, wenn man die Generierung der PDFs nachvollziehen möchte. Daher wird nun beispielhaft die Erzeugung des Nachweises für einen Einzelleistungsnachweis (ELN) der Orthopädie erklärt. Zunächst wird wie bereits angedeutet eine PHP Datei namens *scheindruck.php* aufgerufen. Diese bekommt GET-Parameter überge-

ben, die die für den Druck benötigten Daten übermitteln. Der Aufruf ist beispielhaft in Listing 7.21 aufgeführt.

Listing 7.21: GET-Aufruf der scheindruck.php Datei

```
1 [...]/blauessystem/scheindruck.php?schein=25&scheintyp=1&
    student=11016&semester=21&vorschau=1
```

Die verwendeten Parameter sind *schein*, *scheintyp*, *student*, *semester*, *vorschau*. Der Parameter *schein* übergibt die ID des Scheins der Orthopädie, in diesem Fall 25. *scheintyp* steht für den Scheintyp, der Auswirkung auf die Generierung des Nachweises hat. In diesem Fall ist dem Schein die Zahl 1, sprich Normal, zugeordnet. Die ID des Studenten wird im Parameter *student* übergeben, sodass auf der erzeugten PDF Datei die korrekten Daten des Studenten eingefügt werden. Der Parameter *semester* bezeichnet das ausgewählte Semester, hier 21. Dies entspricht der ID des Semesters, die über die Tabelle *Semester* angesprochen werden kann. Der letzte Parameter *vorschau* gibt in diesem Fall an, dass nur eine Vorschau des Nachweises angezeigt werden soll, der nicht für den Druck bestimmt ist. Aus der *scheindruck.php* Datei wird die *scheindruck()* Funktion in der Datei *scheindruck.inc* aufgerufen. Der weitere Ablauf ist als Ablaufdiagramm in Abbildung 7.2 dargestellt. Diese Übersicht zeigt die Aufrufe von eingebetteten PHP Dateien. In dieser Abbildung fehlen allerdings die Aufrufe, die in derselben Datei stattfinden. Um einen Einblick über den gesamten Ablauf zu erhalten, wird nun der komplette Scheindruck anhand der Abbildung beschrieben.
In der *scheindruck.inc* wird zunächst die *isPrintableScheinStudent()* Funktion aufgerufen, die den Scheintyp, die ScheinID und die StudentID übergeben bekommt. Anhand dieser StudentIDs werden die Studenten auf Scheinkriterien überprüft. Die Funktion *getDetails()* sammelt Scheindaten in einem assoziativen Array. Diese Daten sind der Titel, LPA, Note, Notenzusätze, Datum, Halbjahr, Dozent und dessen Signatur.
Anschließend wird *getStudentDetails()* und *getScheinDetails()* intern aufgerufen, um zum einen die Daten des Studenten zu ermitteln. Zum anderen werden Scheintyp, Scheinvorlage und die Sortierung des übergebenen Scheins ebenfalls in einem Array gespeichert. Diese Daten werden anschließend in ein einziges Array zusammengeführt und an die Funktion *generiereScheine()* übergeben. Diese erstellt zunächst ein PDF Objekt und übergibt den zu druckenden Datensatz an die Methode *einzelschein05()*. Hier wird schließlich der Datensatz ausgelesen und mit dem erzeugten PDF Objekt die Daten in die PDF geschrieben. Anschließend wird in der *scheindruck.inc* das PDF Dokument finalisiert und ausgegeben.
Der hier vorgestellte Scheindruck für einen Einzelleistungsnachweis stellt eine ver-

einfachte Form des Scheindrucks dar. Je nach Nachweis werden unterschiedliche Vorlagen, hier als inc Dateien, eingebunden. Umfassendere Übersichten rufen beispielsweise weitere Funktionen aus der *notenuebersicht.inc* auf. Der Druck des Orthopädie-Scheins soll an dieser Stelle aber als Beispiel genügen.

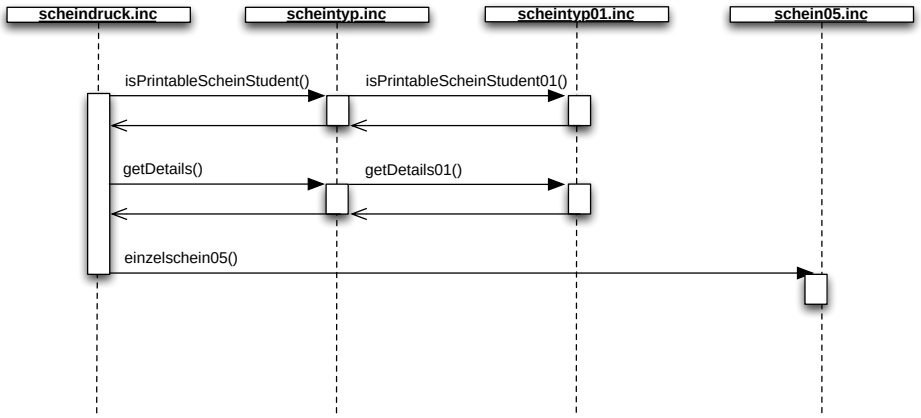


Abbildung 7.2: Ablaufdiagramm für den Druck des Orthopädie-Scheins

7.5.2 Studentenansicht

Die Studentenansicht ist zwar einfacher gestrickt als der Druck von Nachweisen, soll aber an dieser Stelle erklärt werden, hier am Beispiel der *Übersicht Vorklinik*. Zunächst wird per GET-Aufruf der *stud03.php* die Parameter *id* und *seitenmodus* übergeben. Der Aufruf ist in Listing 7.22 aufgeführt. Der erste Parameter repräsentiert die ID des Studenten, der zweite den aktuell ausgewählten Reiter, hier die Übersicht Vorklinik.

Listing 7.22: GET-Aufruf der stud03.php Datei

```
1 [...]/blauessystem/stud03.php?id=12066&seitenmodus=1
```

Eine Übersicht der Aufrufe ist in Abbildung 7.3 zu sehen. Anhand der StudentID wird mit der Funktion *setPaneInfo()* im ersten Schritt das *TabbedPane* erstellt, welches eine Kurzübersicht der studentischen Daten und die Menü-Reiter enthält. Diese Daten werden an das Smarty Template *stud03-01.tpl* weitergegeben. Anschließend wird die *notenuebersicht()* aufgerufen, die die StudentID und den Abschnitt (hier Vorklinik) übergeben bekommt. Hier werden aus der Datenbank mithilfe der StudentID alle Stammdaten des Studenten extrahiert. Des Weiteren wird eine Liste aller Scheine, die der Vorklinik und der entsprechenden Studienordnung zugeordnet sind, abgefragt. Aus den so gesammelten Daten werden in der stud03.php die

Scheine, inklusive der Prüfungsleistungen extrahiert, und ebenfalls an das Smarty Template übergeben.
In der Template Datei selbst werden weitere Template Dateien eingebunden. Die Templates *htmlheader.tpl*, *pageheader.tpl*, *navigation.tpl*, *pagefooter.tpl* und *htmlfooter.tpl* werden standardmäßig in den Template Dateien eingebunden und reduzieren damit den Umfang jeder neu erstellten Template Datei. Die von der *tabbedpane.inc* übergebenen Daten werden in der eingebundenen *tabbedpane.tpl* verarbeitet. Die Übersicht der Scheine wird in der Funktion *leistungsuebersicht()* übernommen, die hier im Template als Smarty Plugin eingebunden wird. In der Template Datei werden dafür Parameter übergeben, die anschließend ausgelesen werden. Dadurch kann in der Template Datei auf tiefergehende Logik verzichtet werden. Den Aufbau der HTML-Tabelle der Scheinübersicht übernimmt die Funktion. Das Template bekommt schließlich von ihr einen String zurück, der die erstellte HTML-Tabelle enthält.

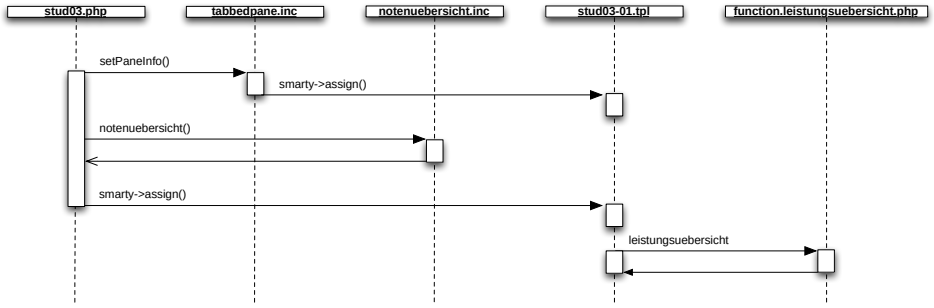


Abbildung 7.3: Ablaufdiagramm für die Anzeige der Übersicht Vorklinik

7.5.3 Studienordnung

Die Studienordnungen wurden bei der Erstentwicklung von Hand in die Datenbank eingetragen und auf eine Dokumentation verzichtet. Daher wird nun das Zusammenspiel der Tabellen rund um die Studienordnung vorgestellt. Abbildung 7.4 zeigt ein Datenbankschema der beteiligten Relationen.

StudienordnungKriterien

Die in dieser Arbeit neu erstellte Relation *StudienordnungKriterien* weist einem Schein und einer Studienordnung ein Scheinkriterium zu. Dieses Kriterium ist wiederum Platzhalter für die ID eines Scheins.

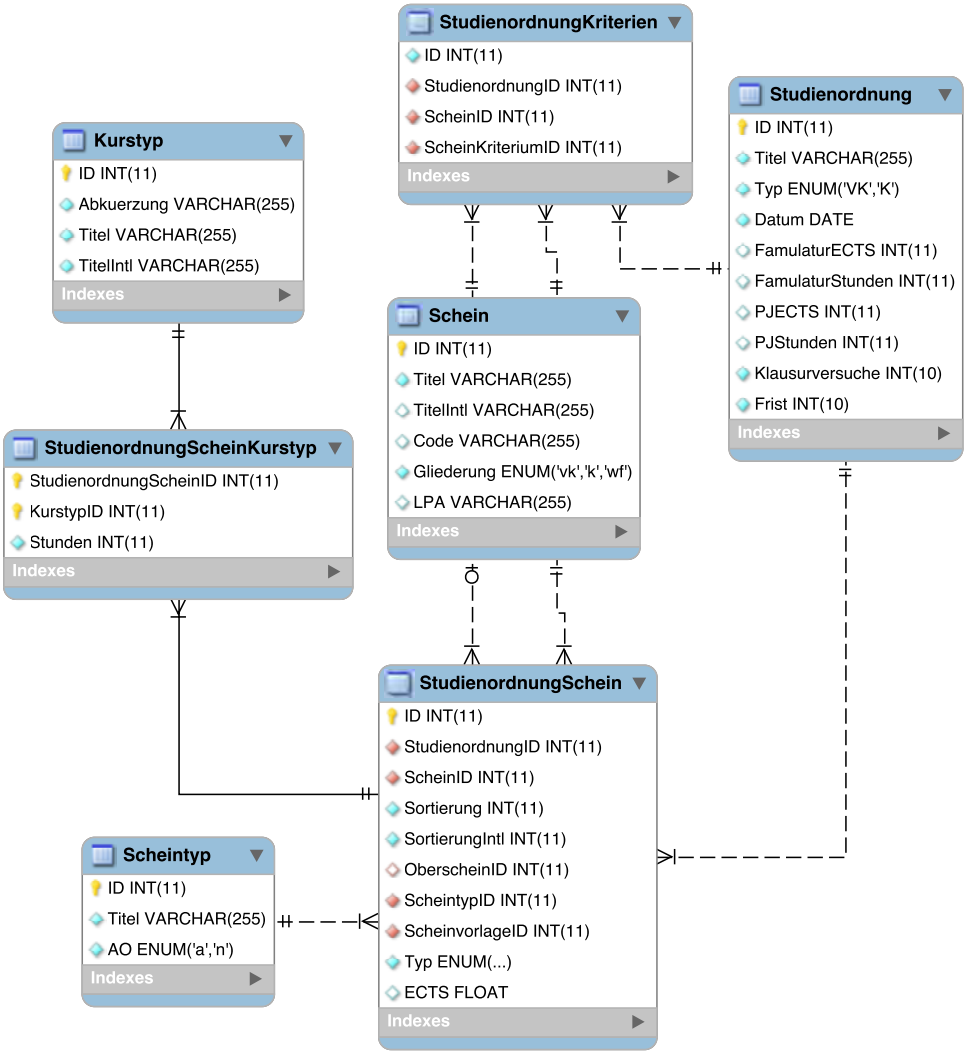


Abbildung 7.4: Datenbankschema der beteiligten Tabellen

StudienordnungSchein

Einer Studienordnung kann eine Anzahl an Scheinen zugeordnet werden. Diese Zuordnung geschieht über das Attribut StudienordnungID. Scheine werden über die *ScheinID* erfasst. Zusätzlich wird ein Scheintyp zugeordnet, der bei der Erstellung von Nachweisen oder Bescheinigungen darüber entscheidet, welche Druckvorlage verwendet wird.

StudienordnungScheinKurstyp

Diese Zwischenrelation ordnet einem StudienordnungSchein einen oder mehrere Kurstypen zu. Dieser wird bei der Ausgabe der Leistungsübersicht im Transcript

of Records angegeben. Der Kurstyp gibt an, ob es sich dabei um eine Vorlesung, einen Kurs/Praktikum, ein Seminar oder ein Stations-/OP-Praktikum handelt.

7.6 Verbesserungsbedarf

Das Blaue System ist funktional, und sofern man sich einen Überblick über das System verschafft hat, in einigen Punkten einfach gestaltet. Der grundsätzliche Aufbau ist immer derselbe. Die Auswahl eines Menüpunkts führt zum Aufruf einer PHP Datei. Diese bindet optional weitere include-Dateien ein und übergibt die Daten anschließend an ein Smarty Template. Allerdings können einige der Funktionen hinterfragt werden, beziehungsweise weisen sie konkreten Verbesserungsbedarf auf. Nachfolgend werden daher einige Themen diesbezüglich angesprochen.

7.6.1 PDF-Erzeugung

Die Erzeugung von PDFs ist ein komplexes Konstrukt aus eingebundenen include Dateien verschiedenster Arten und der Verwendung der TCPDF Library, ausgelagert in mehreren Funktionen und Dateien. Um die Druck-Funktion in ihrer jetzigen Form nachvollziehen zu können, bleibt dem Entwickler keine andere Möglichkeit, als den Ablauf Schritt für Schritt zu debuggen. Je nach Nachweis werden unterschiedliche Dateien inkludiert, beziehungsweise weitere eingebunden. Die Auslagerung von Funktionen und somit die Reduktion von Komplexität sollte nur bis zu einem gewissen Komplexitätsgrad durchgeführt werden. Ein Lösungsansatz wäre, die gesamte Druckfunktion nochmals in ihren verschiedenen Ausführungen auf Papier zu bringen, und eine Neustrukturierung der Funktionalitäten vorzunehmen. Eventuell würde hier ein objektorientierter Ansatz, wie in Abschnitt 7.6.4 beschrieben, die Lesbarkeit des Codes und dementsprechend die Weiterentwicklung und das Debugging vereinfachen.

7.6.2 Erstellung und Zuordnung von Kurstypen

Wie bereits in Abschnitt 7.5.3 vorgestellt, ist die Zuordnung von Kurstypen zu einem *StudienordnungSchein* für die englischsprachige Leistungsübersicht möglich. Bisher wurde diese Zuordnung allerdings initial von Hand in der Datenbank vorgenommen. Im Zuge dieser Arbeit können ab sofort Studienordnungen kopiert und neu erstellt werden. Beim Kopieren werden auch alle Zuordnungen der Zwischenrelation *StudienordnungScheinKurstyp* mitkopiert. Wird allerdings einer Studien-

ordnung ein neuer Schein zugeordnet, fehlt diese Zuordnung zu einem Kurstypen. Hierfür müssten noch Funktionen zur Verwaltung von Kurstypen und deren Zuordnungen zu Studienordnungen realisiert werden.

7.6.3 Benutzerverwaltung

Der Zugriff außerhalb des medizinischen Verwaltungsnetzes ist zwar durch eine *htaccess* Datei gesperrt, aber intern haben Mitarbeiter mit Benutzerkonto vollen Zugriff auf die komplette Funktionalität. Hier sollte die Datenbank entsprechend erweitert werden, um den Zugriff einzelner Benutzer einzuschränken.

7.6.4 Objektorientierung im Blauen System

Im Zuge von PHP 5 wurde auch die Objektorientierung weiter vorangetrieben. Nun ist es unter anderem auch möglich, Variablen und Funktionen in Objekten zu kapseln. Für die Entwicklung, aber auch für die Übersicht des Blauen Systems hat die objektorientierte Programmierung mehrere Vorteile. Durch die Verwendung von Objekten kann die Komplexität des Projektes reduziert werden. Sollen nur Teilbereiche erweitert werden, kann der Entwickler das Objekt instanziieren und die erforderlichen Methoden verwenden, ohne sich weiter in den Code einarbeiten zu müssen. Methoden können somit direkt Objekten zugeordnet werden und müssen nicht in den zahlreichen eingebundenen Dateien gesucht werden. Dadurch würden die Aufrufe auch zumeist selbsterklärend werden. Weitere Kontrolle über die Codestruktur kann über die Definition von Interfaces erreicht werden. Außerdem können seit PHP 5 auch Variablen und Methoden als *private* deklariert und somit vor äußeren Zugriffen, beziehungsweise Änderungen, geschützt werden. Dadurch reduziert sich auch die Fehleranfälligkeit.

7.6.5 Weiterer Ansatz zur Optimierung

Zur Verbesserung der Flexiblität und der Abläufe, ist der Einsatz von Workflows denkbar. Die Arbeit mit dem Blauen System, beziehungsweise die Kommunikation zwischen dem Blauen System und den in dieser Arbeit beschriebenen Systemen, kann auf bestimmte Abläufe reduziert und als Workflows abgebildet werden [52].

7.7 Fazit

In diesem Kapitel wurden Implementierungsdetails vorgestellt, um einen tieferen Einblick in das System zu gewinnen. Wie aufgezeigt, weist das System an einigen Stellen noch Verbesserungsbedarf bezüglich der Programmierung auf. Verbesserungs- und Erweiterungsbedarf gibt es auch hinsichtlich zusätzlicher Funktionen. Dieser Bedarf wird im übernächsten Kapitel 9 im Ausblick ausführlicher diskutiert. Zunächst erfolgt ein Abgleich, inwieweit die an diese Arbeit gestellten Anforderungen erfüllt wurden.

8 Abgleich der Anforderungen

Die Anforderungen aus Kapitel 5, die an diese Arbeit gestellt wurden, sollen nun abgeglichen werden. Inwieweit konnten die Anforderungen umgesetzt und wo mussten Abstriche gemacht werden? Dafür werden die Anforderungen nochmals aufgelistet und erklärt, inwieweit diese erfüllt wurden.

8.1 Blaues System

In diesem Abschnitt werden alle Anforderungen des Blauen Systems aufgeführt und in einer Tabelle dargestellt.

8.1.1 Teilleistungen

Für Veranstaltungen eines Semesters können nun Teilleistungen definiert werden. Hier kann die Anzahl der Teilleistungen festgelegt werden. Zusätzlich kann für eine bessere Zuordnung für jede Teilveranstaltung ein Name vergeben werden. Neben dem Import von Noten ist auch der Import von Punkten möglich, sodass verschiedenste Teilleistungsarten der medizinischen Abteilungen erfasst werden können. Weist ein Student beim Import alle Teilleistungen einer Veranstaltung auf, wird je nach festgelegter Berechnungsart eine Endnote berechnet. Diese Berechnungsart kann bei der Definition der Teilleistungen für die entsprechende Veranstaltung festgelegt werden. Hierbei kann zwischen drei verschiedenen Arten der Berechnung gewählt werden. Beim Import von Noten wird die Endnote über das arithmetische Mittel gebildet. Die Berechnung der Endnote für Punkte geschieht über eine Kumulierung der Punktzahl. Hier wird eine Veranstaltung laut Studienordnung bei einem Prozentsatz der kumulierten Punkte über 60% als *bestanden* angesehen. Daneben kann auch eine Gewichtung in Notenstufen angegeben werden. Hier kann festgelegt werden, wieviel Prozent der Punkte erreicht werden müssen, um beispielsweise die Note *sehr gut* zu erreichen. In diesem Fall wären mindestens 90% der Punkte erforderlich. Wurde eine Endnote berechnet, wird diese als Note in die entsprechende Veranstaltung eingetragen. Zudem können Teilleistungen nun in

der Studentenansicht eingesehen werden. Dabei können für Teilleistungen Nachweise erstellt werden, sofern die Veranstaltung dafür eine Vorlage aufweist.
Die Anforderungen wurden somit erfüllt.

Teilleistungen	
Definition von Teilleistungen für Veranstaltungen	✓
Anzeige und Druck von Teilleistungen in der Studentenansicht	✓
Definition von Namen für einzelne Teilleistungen	✓
Import von Teilleistungen, ähnlich dem Noten-Import	✓
Berechnung einer Endnote unter Berücksichtigung verschiedener Berechnungsarten	✓

8.1.2 Studienordnung

Studienordnungen können nun kopiert und angepasst werden. Beim Kopieren werden alle Einträge und Attribute der zu kopierenden Studienordnung aus den Tabellen *Studienordnung*, *StudienordnungSchein* und *StudienordnungScheinKurstyp* kopiert und der neu erstellten Studienordnung zugeordnet. Anschließend kann diese, wie auch bereits vorhandene Studienordnungen, bearbeitet und erweitert werden. Bei der Bearbeitung können noch nicht zugewiesene Scheine zugewiesen und bestehende Scheine abgewählt werden. Bei der Zuweisung eines neuen Scheins öffnet sich eine Eingabemaske, in der Attribute wie Sortierung, Oberschein und Scheinvorlage angegeben werden können. Innerhalb der Plausibiltäts-Checks, die nachfolgend beschrieben werden, können den Studienordnungen nun auch Fristen und Klausurversuche zugeordnet werden.
Alle Anforderungen konnten somit erfüllt werden.

Studienordnung	
Erweiterung der Studienordnung um Fristen und Klausurversuche	✓
Definition von Schein-Voraussetzungen	✓
Verwaltung von Studienordnungen	✓

8.1.3 Plausibilitäts-Checks

Im Blauen System können nun, wie gefordert, Plausibilitäts-Checks durchgeführt werden. Hierbei kann im gleichlautenden Menüpunkt zunächst die Voraussetzung

für einen erfolgreichen Abgleich mit Corona geschaffen werden. Dazu können Scheine, beziehungsweise Veranstaltungen im Blauen System mit einer aus Corona importierten Veranstaltungsliste abgeglichen werden. Korrespondierenden Veranstaltungen werden dann LSFIDs zugeordnet. Diese LSFIDs dienen dann der Zuordnung von Teilveranstaltungen aus Corona, in denen jeweils die für die Prüfung angemeldeten Studenten gespeichert sind. Bei jedem Login wird der Datenbestand des Blauen Systems auf Klausur- und Fristüberschreitungen überprüft. Zusätzlich werden Anfragen an Corona gestellt, die die Veranstaltungen des aktuell ausgewählten Semesters überprüfen. Wurden für Veranstaltungen des aktuellen Semesters, wie bereits erwähnt, Zuordnungen getroffen, können anhand der Teilveranstaltungen alle angemeldeten Studenten aus Corona importiert werden. Diese können dann nach erfüllten, beziehungsweise nicht erfüllten Voraussetzungen überprüft werden. Auch diese Voraussetzungen müssen zuvor initial für jede Studienordnung definiert werden.
Die Anforderungen an die Plausibilitäts-Checks sind somit erfüllt.

Plausibilitäts-Checks	
Abbilden von Klausurversuchen und -fristen je nach Studienordnung	✓
Prüfen von Noten & Teilleistungen	✓
Prüfen von Klausuranmeldungen aus Corona	✓
Realisierung von Kurz-Checks	✓
Verbergen von Fehlern unter Angabe von Gründen	✓
CSV-Export von Fehlerlisten	✓

8.1.4 Kohortenansicht

Die Kohortenansicht wurde im System umgesetzt. Anhand des ausgewählten Semesters wird eine Liste an Veranstaltungen in eine Auswahlliste geladen. Für eine ausgewählte Veranstaltung werden dann die teilnehmenden Kohorten angezeigt, die einzeln oder mehrfach auswählbar sind. Bei der Auswahl einer Kohorte wird eine Liste der Studenten angezeigt, sortiert nach vorhandenen und nicht vorhanden Prüfungsleistungen. Noten können hier direkt bearbeitet oder neu eingetragen werden.
Somit wurden alle Anforderungen umgesetzt.

Kohortenansicht	
Ausgabe von Ergebnissen je nach ausgewählter Veranstaltung	✓
Sortierung nach Kohorten und Noten	✓
Direkte Bearbeitung von Noten	✓
Kohorten an- und abwählbar machen	✓

8.1.5 Studentenansicht

Eingetragene Teilleistungen werden nun, wie gefordert, zusätzlich zu den Schein-Noten in der *Übersicht Vorklinik* und *Übersicht Klinik* angezeigt. Zusätzlich wurde die Ansicht um Log-Einträge erweitert. Auch im Reiter *Noten & Scheine* wurde die nach Semestern sortierte Anzeige von Noten um Teilleistungen erweitert. Neben den Log-Einträgen für Teilleistungen werden nun auch Ergebnisse des Plausibilitäts-Checks des ausgewählten Studenten angezeigt, um beispielsweise bei einer Beratung direkt Probleme im Studienverlauf erkennen zu können.
Alle geforderten Funktionen der Studentenansicht wurden erfüllt.

Studentenansicht	
Druck von Bescheinigungen für alle vorhandenen Wahlfächer	✓
Anzeige von Ergebnissen der Plausibilitäts-Checks für jeden Studenten	✓
Anzeige Teilleistung-Logs	✓

8.1.6 PDFs

Ehemals statische Inhalte wie der Ansprechpartner und zusätzliche Hinweise zu Bescheinigungen werden nun dynamisch aus der Datenbank geladen und bei der Erzeugung der PDF-Dateien eingefügt. Außerdem können nun auch für Teilleistungen Nachweise generiert werden, wenn für den zugeordneten Schein eine Vorlage existiert.
Alle Anforderungen wurden erfüllt.

PDFs	
Daten des Ansprechpartners in Bescheinigungen und Nachweisen editierbar machen	✓
Drucken von Bescheinigungen für Vorklinik und Klinik in einer PDF	✓
Drucken von Bescheinigungen für Vorklinik und Klinik für internationale Studenten in einer PDF	✓
Neue Leistungsübersicht für die Anerkennung von Studienleistungen	✓
Info-Text in der Beschreibung des Praktischen Jahres editierbar machen	✓

8.2 Corona

Das kiz möchte Corona vor externen Zugriffen schützen, besonders bei der Programmierung von zusätzlichen Schnittstellen. Daher einigte man sich darauf, dass die erforderlichen Schnittstellen vom kiz, in Person von Herrn Aschoff, bereitgestellt werden. Diese Schnittstellen sind auf bestimmte IPs beschränkt, die sich zusätzlich im Subnetz der Universität Ulm befinden.
Die Anforderungen wurden hier durch die Zusammenarbeit mit dem kiz erfüllt.

Corona	
Import von Veranstaltungen und Teilveranstaltungen aus Corona	✓
Import von Prüfungsanmeldungen anhand Teilveranstaltungs-IDs	✓
Abgleich der importierten Veranstaltungen mit Scheinveranstaltungen aus Blauem System	✓

8.3 MedicUlm

Um auch für Studenten einen Mehrwert zu bieten, ist eine Notenabfrage durch die bereits bestehende *MedicUlm* Android App [43] der Medizinischen Fakultät möglich. Der Benutzer kann sich über ein Loginverfahren mit seinem kiz-Logindaten authentifizieren und anschließend eine Notenansicht erhalten. Diese Abfrage ist nicht manipulierbar und der Server des Blauen Systems nach außen hin anonym, indem die App über eine Zwischenstelle mit dem Blauen System kommuniziert. Ein WebService leitet hierbei die Kommunikation zwischen MedicUlm und Blauem

System, sodass das System aus Sicht des Anwenders im Hintergrund bleibt.
Alle Anforderungen wurden erfüllt.

MedicUlm	
Erweiterung der Menüstruktur um Notenansicht	✓
Login-Screen zur Eingabe von kiz-Logindaten	✓
Aufruf des getGrades-WebService	✓
Parsen und Anzeige der XML-Notenliste	✓

8.4 WebService

Es wurde ein getGrades WebService erstellt, der die Kommunikation zwischen der MedicUlm Android App und dem Blauen System regelt. Dieser gibt die Login-Daten der App an das Blaue System weiter und erhält vom Blauen System im Erfolgsfall eine Notenliste. Diese wird anschließend in der MedicUlm App angezeigt. Die Prüfung der Login-Daten erfolgt durch den in der DBIS bereits vorhandenen auth-WebService.
Alle Anforderungen wurden erfüllt.

WebService	
Erstellung eines WebServices, der kiz-Logindaten an Blaues System weitergibt	✓
Rückgabe der XML-Notenliste aus dem Blauen System als Response-Nachricht	✓

8.5 Moodle

Im Zuge dieser Arbeit sollte eine Kommunikation zwischen der Lernplattform Moodle und Corona ermöglicht werden. Bei der Anmeldung eines Studenten für eine Veranstaltung sollte der entsprechende Moodle Kurs automatisch abonniert werden. Leider kam es durch die in Kapitel 5, Abschnitt 5.8.3 beschriebenen Schwierigkeiten dazu, dass die Moodle-Thematik in dieser Arbeit nicht mehr umgesetzt werden konnte. Nachdem ersichtlich wurde, dass der Zeitplan nicht mehr eingehalten werden konnte, wurde der Schwerpunkt dieser Arbeit darauf gesetzt,

die Anforderungen seitens des Blauen Systems, der MedicUlm App, Corona und des WebServices zu erfüllen.
Die Anforderungen, die an die Erweiterung von Moodle gestellt wurden, wurden nicht erfüllt.

Moodle	
Schnittstelle zwischen Moodle & Corona	✗
Prüfungsanmeldung in Corona führt zum automatischen Abonnieren des Moodle-Kurses	✗
Trotz Automatisierung auch weiterhin manuelles Abonnieren von Kursen	✗

8.6 Fazit

Alle an diese Arbeit gestellten Anforderungen konnten - die Moodle Funktionalität ausgenommen - erfüllt werden. Warum die Moodle Anforderungen nicht mehr umgesetzt werden konnte, wurde bereits in Kapitel 5 im Abschnitt 5.8.3 erklärt. Dort werden die Schwierigkeiten aufgeführt, die zu diesem Umstand führten. Daher ist diese Anforderung auch Teil der Zusammenfassung und des Ausblicks, die im nächsten und damit letzten Kapitel folgen.

9 Zusammenfassung und Ausblick

In diesem Kapitel werden die wichtigsten Punkte dieser Arbeit nochmals zusammengefasst und in einen Zusammenhang gestellt. Anschließend wird noch ein Ausblick darauf gegeben, inwiefern das Blaue System, beziehungsweise die daran angebundenen Systeme, noch verbessert, erweitert oder weiter eingebunden werden können.

9.1 Zusammenfassung

Grundlage dieser Arbeit war eine enge Zusammenarbeit zwischen dem Institut der Datenbanken und Informationssysteme, dem Studiendekanat der Medizinischen Fakultät und dem Kommunikations- und Informationszentrum der Universität Ulm. Diese Zusammenarbeit ermöglichte es auch, einen Großteil der an diese Arbeit gestellten Anforderungen umzusetzen. Wie bereits in Abschnitt 5.8 in Kapitel 5 angesprochen, fiel die Einarbeitung in das Blaue System zunächst schwer. Hintergrundwissen hinsichtlich der medizinischen Studiengänge und deren teils komplexer Struktur musste erarbeitet und in Gesprächen mit den Mitarbeitern des Studiendekanats diskutiert werden. Auch in Bezug auf deren Abbildung im Blauen System. Dieses erforderte zunächst einen höheren Einarbeitungsaufwand als ursprünglich angenommen. Das System wies keinerlei Dokumentation auf. An vielen Stellen gab es keine tiefergehenden Kommentare bezüglich der Programmierlogik. Hinsichtlich der Anforderungen war die Vorlage des Studiendekanats ein Wegweiser für diese Arbeit. Diese Anforderungen wurden in vielen Kleingruppen-Gesprächen mit den beiden Ansprechpartnern des Studiendekanats diskutiert und im Detail weiter ausgeführt. Auf Grundlage dieser Gespräche wurden die Anforderungen implementiert und unter stetigen Rückmeldungen an die Bedürfnisse der Mitarbeiter angepasst. Im Plenum erwies sich die eine oder andere Funktion allerdings als nicht ausreichend, beziehungsweise für die Zwecke der anderen Mitarbeiter als völlig unpassend. Diese Gründe sorgten dafür, dass der Zeitplan dieser Arbeit angepasst werden musste. Bei den an diese Arbeit gestellten Anforderungen, mussten bezüglich der Moodle Schnittstelle Abstriche gemacht werden. Diese Anforderung ist somit auch Teil des Ausblicks. Abgesehen davon wurden in dieser

Arbeit die restlichen Anforderungen in der Hinsicht erfüllt, dass sie den täglichen Arbeitsaufwand der Mitarbeiter reduzieren und den Arbeitsalltag erleichtern, beziehungsweise ihnen Werkzeuge an die Hand geben, die eine noch umfassendere Verwaltung ermöglichen.

9.2 Ausblick

Es bleibt festzustellen, dass das Blaue System im Zuge dieser Arbeit um viele Funktionalitäten erweitert wurde. Davon abgesehen gibt es am bestehenden System selbst noch Verbesserungsbedarf. Auch konnte die Anforderung hinsichtlich der Schnittstelle zwischen der Lehrplattform Moodle und der Anmeldungsverwaltung Corona nicht erfüllt werden. Die MedicUlm App wurde zwar um die Notenansicht erweitert, allerdings muss hier noch abgewägt werden, inwiefern der Datenschutz und Sicherheitsaspekte eingehalten wurden. Ein Schritt wäre der Aufruf, beziehungsweise die Einführung, einer sicheren SSL-Verbindung zwischen App, WebService und dem Blauen System. Diese Thematiken wären Teil einer möglichen weiteren Arbeit, beziehungsweise eines nachfolgenden Projektes, das aufgrund der positiven Erfahrungen abermals zwischen der DBIS-Abteilung und dem Studiendekanat, denkbar wäre.
Abgesehen von noch zu verbessernden und umzusetzenden Funktionen stellt sich die Frage, inwieweit das Blaue System für den Bedarf des Studiendekanats geeignet ist. In Kapitel 2 wurden ähnliche, beziehungsweise im Hochschulsektor konkurrierende Verwaltungssysteme vorgestellt. All diesen Systemen ist gemein, dass sie einen größeren Leistungsumfang als das Blaue System aufweisen. Funktionen, die im Zuge einer verbesserten Verwaltung im Laufe der Zeit noch gefordert werden, sind dort möglicherweise schon vorhanden. Davon abgesehen wäre ein Umstieg mit einem hohen Kostenfaktor verbunden. Dies zöge unter anderem Anforderungsanalysen, Mitarbeitergespräche, Installation von Software und Servern und konstante Wartungskosten nach sich. Dem spricht entgegen, dass das Blaue System bereits direkt an die Bedürfnisse des Studiendekanats angepasst ist und sich im Laufe der Jahre bewährt hat. Das System erfordert zwar eine gewisse Einarbeitungszeit, ist allerdings einfacher erweiterbar als ein komplexes, kommerzielles Framework. Dies hat den Vorteil, dass keine externen Mitarbeiter-Ressourcen nötig sind, sondern Implementierungsarbeiten von Studierenden im Zuge von Projekten oder Arbeiten erledigt werden können.
Zusammenfassend kann festgestellt werden, dass das Blaue System kein fertiges Verwaltungssystem darstellt, sondern wie die meisten tagtäglich verwendeten Systeme, stetigen Verbesserungsbedarf aufweist. Das System mag zwar nicht den

deutlich höheren Umfang von kommerziellen Lösungen aufweisen, bietet dadurch allerdings ein übersichtliches System, das in vielerlei Hinsicht einfach zu verstehen und zu erweitern ist. Solange seitens des Studiendekanats der Umfang eines kommerziellen Systems nicht gewünscht ist, beziehungsweise dieser Umfang durch vorhandene Systeme geleistet wird, genügt das Blaue System den Ansprüchen an eine Prüfungsverwaltung völlig.

Anhang

Functions

Function Summary

void	abbrechen()
void	aktualisiereFehlerAnzeige()
void	anschriftzeile(mixed inhalt, bool emph, bool indent, int width) anschriftzeile() erzeugt eine Zeile der Anschrift $value: Inhalt der Zeile $emph: Fettdruck? $indent: Soll das Feld eingerueckt werden? $width: Breite des Feldes
int	arithmetischesMittel(Array teilleistungen, Array berechnungsDetails, String aktTeilleistungName)
void	bearbeiten()
void	benutzerAnlegen(mixed modus)
void	benutzerAnzeigen()
void	benutzerBearbeiten(mixed id, mixed useDB, mixed modus)
void	berechneGesamtnote($aktTeilleistungName die, mixed aktVeranstaltung, mixed studentID, mixed aktTeilleistungName)
void	besitztLock(mixed id, mixed art, mixed benutzer) besitztLock() ueberprueft ob der Benutzer noch einen gueltigen Lock auf den Datensatz besitzt $id: die ID des Datensatz $art: die Art des Locks $benutzer: die ID des bearbeitenden Benutzers
void	checkCoronaVeranstaltungen()
void	checkFristKlausurversuche(Array erg,, String system, boolean checked, mixed erg)
void	checkTeilleistungen(Array verg, mixed system, mixed checked)
void	checkVoraussetzungen(String matrikelListe, String aktCorona, mixed matrikelArray, mixed system, int checked)
void	cmp_scheinDaten(mixed a, mixed b)
void	cmp_scheinDatenTL(mixed a, mixed b)
void	cmp_scheinansicht(mixed a, mixed b) Sortiere zunaechst nach Grupperiung (ELN,BP,...), dann nach Sortierung laut SO-Schein-Reihenfolge
void	cmp_scheine(mixed a, mixed b)
void	cmp_scheineTL(mixed a, mixed b)
void	cmp_suggestions(mixed a, mixed b)
void	computeAverage(mixed data, int count)
void	computeDeviation(mixed data, mixed avg, int count)
void	datensatzExistiert(mixed id, mixed art) datensatzExistiert() ueberprueft die Existenz des mittels $art angegebenen Datensatzes $id: ID des zu ueberpruefenden Datensatzes $art: Art des Datensatzes, z.B.

void	datensatzGesperrt(mixed id, mixed art) datensatzGesperrt() prueft, ob ein Datensatz gesperrt ist.
void	defaults()
void	delete()
void	dozentAnlegen(mixed modus)
void	dozentBearbeiten(mixed id, mixed content, mixed modus)
void	dozentenAnzeigen()
void	edit()
void	eintraegeProTag(mixed tabelle, mixed zeitpunktBeginn, mixed s_von, mixed s_bis, mixed s_avg, mixed s_dev)
void	eintragungLoeschen()
void	einzelschein04(mixed daten)
void	einzelschein05(mixed daten)
void	einzelschein06(mixed daten)
void	einzelschein07(mixed daten)
void	einzelschein08(mixed daten)
void	einzelschein09(mixed daten)
void	einzelschein10(mixed daten)
void	einzelschein11(mixed daten)
void	einzelschein12(mixed daten)
void	einzelschein14(mixed daten)
void	erstelleScheinNotenlisten(mixed studienordnungen, mixed abschnitt, mixed art, mixed daten, int start, int limit, int v, bool gruppierung) Sucht fuer alle uebergebenen Studienordnungen die entsprechenden Scheinlisten (inkl.
void	erstelleScheinNotenlistenQuerformat(mixed studienordnungen, mixed abschnitt, mixed art, mixed daten, int start, int limit, int v, bool gruppierung) Sucht fuer alle uebergebenen Studienordnungen die entsprechenden Scheinlisten (inkl.
void	erstelleXMLNotenListe(String matrikelnummer) Erstellt anhand der Matrikelnummer eine XML Liste, die vom Android ausgelesen wird.
void	fehler()
void	formFehler(mixed typid, mixed typ, mixed feldid, mixed fehlertext, mixed previousSeitenmodus) formFehler() zeigt Smarty an, dass es bei einer Formulareingabe einen Fehler gegeben hat $typid: ID des Typs $typ: Typ der Verwaltungsseite $feldid: ID des Formfeldes, das den Fehler enthaelt $fehlertext: anzuzeigender Fehlertext $previousSeitenmodus: wenn TRUE, auf letzten Seitenmodus zuruecksetzen
void	generiereScheine(mixed scheinDaten, mixed uebersicht) generiereScheine($scheinDaten, $uebersicht) $scheinDaten enthaelt fuer jede Student/Schein-Kombination folgendes Array: (scheinid, vorlage, vorschau, studentid, matrikel, name, vorname, geburtsdatum, geburtsort, titel, lpa, note, noten, von, bis, datum, halbjahr, dozenten, sig)
void	generiereScheineTL(mixed scheinDaten, mixed uebersicht) generiereScheine($scheinDaten, $uebersicht) $scheinDaten enthaelt fuer jede Student/Schein-Kombination folgendes Array: (scheinid, vorlage, vorschau,

	studentid, matrikel, name, vorname, geburtsdatum, geburtsort, titel, lpa, note, noten, von, bis, datum, halbjahr, dozenten, sig)
void	generiereUebersicht(mixed daten, mixed art, mixed header, mixed vorschau, bool return) Gibt eine Notenuebersicht des oder der Studenten aus $daten: Array mit allen zu druckenden Daten Aufbau: Studentenweise mit folgenden Feldern: ID, Name, Vorname, Matrikel, Strasse, PLZ, Ort, Geburtsort, Geburtsdatum, Famulatur, Datum, Scheine[VK/K] => Array(Scheintyp, Titel, LPA, Note, Datum) $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $header: Gibt an, ob ein pdf-Header erstellt werden soll $vorschau: Gibt an, ob es sich um eine Druckvorschau handelt $return: Gibt an, wie die PDF-Datei zurueck gegeben werden soll TRUE: Rueckgabe String (fuer Notenspiegel); FALSE: Rueckgabe PDF
void	generiereUebersichtQuerformat(mixed daten, mixed art, mixed header, mixed vorschau, bool return) Gibt eine Notenuebersicht des oder der Studenten aus $daten: Array mit allen zu druckenden Daten Aufbau: Studentenweise mit folgenden Feldern: ID, Name, Vorname, Matrikel, Strasse, PLZ, Ort, Geburtsort, Geburtsdatum, Famulatur, Datum, Scheine[VK/K] => Array(Scheintyp, Titel, LPA, Note, Datum) $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $header: Gibt an, ob ein pdf-Header erstellt werden soll $vorschau: Gibt an, ob es sich um eine Druckvorschau handelt $return: Gibt an, wie die PDF-Datei zurueck gegeben werden soll TRUE: Rueckgabe String (fuer Notenspiegel); FALSE: Rueckgabe PDF
void	geschlechtsUmwandlung(mixed geschlecht) geschlechtsUmwandlung() aendert den String 'männlich' bzw.
void	getAllDozenten() Sucht alle vorhandenen Dozenten aus der Datenbank und liefert diese zurueck
void	getAllDozentenSignaturen(mixed dozenten) Durchsucht das Signatur-Verzeichnis nach Unterschriften der uebergebenen Dozenten und schreibt das Ergebnis (Signatur vorhanden ja/nein) in das Rueckgabe-Array
void	getAllScheine() Sucht alle vorhandenen Scheine aus der Datenbank und gruppiert diese nach VK/K/WF.
void	getAllSemester(bool auchOhneVeranstaltungen) Sucht alle vorhandenen Semester mit zugehoeriger Anzahl von Veranstaltungen aus der Datenbank.
void	getArt()
void	getBenutzer()
void	getBenutzerName()
void	getBesteVeranstaltung(mixed scheintyp, mixed schein, mixed studenten) Liefert die beste Veranstaltung fuer jeden Studenten zurueck, die der Student fuer diesen Schein besucht hat.
void	getBesteVeranstaltung01(mixed schein, mixed studenten, bool zweites_wahlfach)
void	getBesteVeranstaltung02(mixed schein, mixed studenten)

void	getBesteVeranstaltung03(mixed schein, mixed studenten)
void	getBesteVeranstaltung04(mixed schein, mixed studenten, bool zweites_wahlfach)
void	getBesteVeranstaltung05(mixed schein, mixed studenten)
void	getBesteVeranstaltung07(mixed schein, mixed studenten)
void	getCounterScheinSemester(mixed scheintyp, mixed schein, mixed semester, bool scheintypOrigin) getCounterScheinSemester($scheintyp, $schein, $semester) $scheintyp: der Typ des Scheins $schein: die ID des Scheins $semester: die ID des Semesters returns Array (TRUE, gedruckt, insgesamt) wenn Studenten in Veranstaltungen und nicht in LogScheindruck eingetragen sind Array (FALSE, gedruckt, insgesamt) wenn Studenten in Veranstaltungen und in LogScheindruck eingetragen sind
void	getCounterScheinSemester01(mixed schein, mixed semester, mixed scheintypOrigin)
void	getCounterScheinSemester02(mixed schein, mixed semester, mixed scheintypOrigin)
void	getCounterScheinSemester03(mixed schein, mixed semester, mixed scheintypOrigin)
void	getCounterScheinSemester04(mixed schein, mixed semester, mixed scheintypOrigin)
void	getCounterScheinSemester05(mixed schein, mixed semester, mixed scheintypOrigin)
void	getCounterScheinSemester07(mixed schein, mixed semester, mixed scheintypOrigin)
void	getDatum(mixed scheintyp, mixed schein, mixed studenten, mixed typ) Liefert das Datum (von, bis, datum) des Scheins fuer jeden uebergebenen Studenten zurueck.
void	getDatum01(mixed schein, mixed studenten, mixed typ)
void	getDatum02(mixed schein, mixed studenten, mixed typ)
void	getDatum03(mixed schein, mixed studenten, mixed typ)
void	getDatum04(mixed schein, mixed studenten, mixed typ)
void	getDatum05(mixed schein, mixed studenten, mixed typ)
void	getDatum07(mixed schein, mixed studenten, mixed typ)
void	getDays(mixed date1, mixed date2)
void	getDetails(mixed scheintyp, mixed schein, mixed studenten) Liefert die Details (Titel, LPA-Code, Note, Untertitel, Datum, Halbjahr, Dozenten, Signaturen) zurueck.
void	getDetails01(mixed schein, mixed studenten, bool zweites_wahlfach)
void	getDetails02(mixed schein, mixed studenten)
void	getDetails03(mixed schein, mixed studenten)
void	getDetails04(mixed schein, mixed studenten, bool zweites_wahlfach)
void	getDetails05(mixed schein, mixed studenten)
void	getDetails07(mixed schein, mixed studenten)
void	getDistinctUnterscheine02(mixed schein)
void	getDozent(mixed scheintyp, mixed schein, mixed studenten) Liefert die Dozenten des Scheins fuer jeden uebergebenen Studenten zurueck.
void	getDozent01(mixed schein, mixed studenten)
void	getDozent02(mixed schein, mixed studenten)

void	getDozent03(mixed schein, mixed studenten)
void	getDozent04(mixed schein, mixed studenten)
void	getDozent05(mixed schein, mixed studenten)
void	getDozent07(mixed schein, mixed studenten)
void	getHalbjahr(mixed scheintyp, mixed schein, mixed studenten) Liefert die Werte 1, 2 oder 3 zurueck, mit denen die Halbjahre angezeigt werden, in denen der Student die Veranstaltungen fuer diesen Schein besucht hat.
void	getHalbjahr01(mixed schein, mixed studenten)
void	getHalbjahr02(mixed schein, mixed studenten)
void	getHalbjahr03(mixed schein, mixed studenten)
void	getHalbjahr04(mixed schein, mixed studenten)
void	getHalbjahr05(mixed schein, mixed studenten)
void	getHalbjahr07(mixed schein, mixed studenten)
void	getJahr()
void	getKohortenAnzahlViaVeranstaltung(String aktVeranstaltung)
void	getLPA(mixed scheintyp, mixed schein) Liefert den LPA-Code des Scheins zurueck.
void	getLPA01(mixed schein)
void	getLPA02(mixed schein)
void	getLPA03(mixed schein)
void	getLPA04(mixed schein)
void	getLPA05(mixed schein)
void	getLPA07(mixed schein)
void	getLeistungsuebersichtdaten(mixed abschnitt, mixed kohorten, mixed sortierung)
void	getNavVisible()
void	getNavigation()
void	getNote(mixed scheintyp, mixed schein, mixed studenten, mixed typ) Liefert die Note des Studenten fuer diesen Schein zurueck.
void	getNote01(mixed schein, mixed studenten, mixed typ, bool zweites_wahlfach)
void	getNote02(mixed schein, mixed studenten, mixed typ)
void	getNote03(mixed schein, mixed studenten, mixed typ)
void	getNote04(mixed schein, mixed studenten, mixed typ, bool zweites_wahlfach)
void	getNote05(mixed schein, mixed studenten, mixed typ)
void	getNote07(mixed schein, mixed studenten, mixed typ)
void	getNotenzusatz(mixed scheintyp, mixed schein, mixed studenten, mixed typ) Liefert den Notenzusatz, falls vorhanden, des Studenten fuer diesen Schein zurueck.
void	getNotenzusatz01(mixed schein, mixed studenten, mixed typ)
void	getNotenzusatz02(mixed schein, mixed studenten, mixed typ)
void	getNotenzusatz03(mixed schein, mixed studenten, mixed typ)
void	getNotenzusatz04(mixed schein, mixed studenten, mixed typ)
void	getNotenzusatz05(mixed schein, mixed studenten, mixed typ)
void	getNotenzusatz07(mixed schein, mixed studenten, mixed typ)

void	getNotizArten()
void	getPJWahlfach() Liefert die vorhandenen PJ-Wahlfaecher fuer die Studenten-Verwaltungsseiten zurueck (inkl.
void	getPOName(integer aktPO)
void	getParam(mixed name, mixed method) getParam() gibt einen Aufrufparameter des Skriptes zurueck Reihenfolge des Ueberpruefung: GET, POST $name: Name des Parameters $method: Uebertragungsmethode (GET, POST, beides)
void	getPostSeitenmodus() getPostSeitenmodus() ueberprueft, ob ein Parameter der Form 'seitenmodus_xx' uebergeben wurde, und gibt den Teil 'xx' zurueck.
void	getPreviousSeitenmodus()
void	getSO() getSO() sucht alle vorhandenen Studienordnungen, gliedert sie in VK und K und gibt diese beiden Arrays zurueck
void	getScheinDetails(mixed schein, mixed studenten)
void	getScheinDetailsTL(mixed schein, mixed studenten)
void	getScheinSemester(mixed scheintyp, mixed schein, mixed studenten) Einfach nur das Semester zurückliefern.
void	getScheinVollstaendigkeit(mixed studentid, mixed abschnitt)
void	getScheine(mixed semester)
void	getScheineBestanden(mixed studentid, mixed scheine)
void	getScheineSO(mixed studentid, mixed type)
void	getScheineTL(mixed semester)
void	getSeitenmodus() getSeitenmodus() gibt den uebergebenen Seitenmodus zurueck, oder SEITENMODUS_DEFAULT falls keiner gesetzt wurde.
void	getSemester()
void	getSignatur(mixed scheintyp, mixed schein, mixed studenten) Liefert die Dateinamen der Signaturen fuer jeden uebergebenen Studenten zurueck.
void	getSignatur01(mixed schein, mixed studenten)
void	getSignatur02(mixed schein, mixed studenten)
void	getSignatur03(mixed schein, mixed studenten)
void	getSignatur04(mixed schein, mixed studenten)
void	getSignatur05(mixed schein, mixed studenten)
void	getSignatur07(mixed schein, mixed studenten)
void	getStudentDetails(mixed studenten)
void	getStudentDetailsTL(mixed studenten)
void	getStudentIDs(mixed scheintyp, mixed schein, mixed semester, mixed logscheindruck, bool studenten, bool scheintypOrigin)
void	getStudentIDs01(mixed schein, mixed semester, mixed logscheindruck, bool studenten, mixed scheintypOrigin)
void	getStudentIDs02(mixed schein, mixed semester, mixed logscheindruck, bool studenten, mixed scheintypOrigin)
void	getStudentIDs03(mixed schein, mixed semester, mixed logscheindruck, bool studenten, mixed scheintypOrigin)

void	getStudentIDs04(mixed schein, mixed semester, mixed logscheindruck, bool studenten, mixed scheintypOrigin)
void	getStudentIDs05(mixed schein, mixed semester, mixed logscheindruck, bool studenten, mixed scheintypOrigin)
void	getStudentIDs07(mixed schein, mixed semester, mixed logscheindruck, bool studenten, mixed scheintypOrigin)
void	getStudentenNotenViaKohorten(String selected_kohorten, mixed aktVeranstaltung)
void	getStudentenNotenViaKohortenOhne(String selected_kohorten, mixed aktVeranstaltung)
void	getSuche()
void	getTitel(mixed scheintyp, mixed schein, mixed studenten, mixed typ) Liefert den Titel des Scheins fuer jeden uebergebenen Studenten zurueck.
void	getTitel01(mixed schein, mixed studenten, mixed typ, str teilleistung)
void	getTitel02(mixed schein, mixed studenten, mixed typ)
void	getTitel03(mixed schein, mixed studenten, mixed typ)
void	getTitel04(mixed schein, mixed studenten, mixed typ, bool zweites_wahlfach)
void	getTitel05(mixed schein, mixed studenten, mixed typ)
void	getTitel07(mixed schein, mixed studenten, mixed typ)
void	getUnterscheine(mixed scheintyp, mixed schein) Liefert die Unterscheine (falls vorhanden) fuer diese Schein-/Scheintyp-Kombination zurueck
void	getUnterscheine01(mixed schein)
void	getUnterscheine02(mixed schein)
void	getUnterscheine03(mixed schein)
void	getUnterscheine04(mixed schein)
void	getUnterscheine05(mixed schein)
void	getUnterscheine07(mixed schein)
void	getUntertitel(mixed scheintyp, mixed schein, mixed studenten) Liefert die Untertitel des Scheins fuer jeden uebergebenen Studenten zurueck (z.B.
void	getUntertitel01(mixed schein, mixed studenten)
void	getUntertitel02(mixed schein, mixed studenten)
void	getUntertitel03(mixed schein, mixed studenten)
void	getUntertitel04(mixed schein, mixed studenten)
void	getUntertitel05(mixed schein, mixed studenten)
void	getUntertitel07(mixed schein, mixed studenten)
void	getVeranstaltungTitel(integer aktVeranstaltung)
void	getVeranstaltungen05(mixed schein, mixed studenten)
void	getVeranstaltungsListe()
void	getVersion() getVersion() gibt die Versionsnummer zurueck
void	globalConfig(mixed title, mixed vlabel)
void	gueltigeSession() Prueft ob in der aktuellen Session eine BenutzerID abgelegt ist und diese in der Datenbank noch existiert
void	hasChild(mixed p)

void	hasParams(mixed name, mixed quantifier, mixed method) hasParams() testet auf die Existenz aller Aufrufparameter / mindestens eines Aufrufparameters $name: Array mit den Namen der Parameter $quantifier: ANY/ALL Ueberpruefung auf einen/alle Parameter $method: Uebertragungsmethode (GET, POST, beides)
void	headerNotenliste() headerNotenliste() ist ein Wrapper zum Erstellen der Kopfzeile einer Notenliste
void	hideFehler(integer errorid, String checkBS)
void	holeAlleSemester()
void	holeKlausurversucheFrist(mixed po)
void	holeKriterien(Array scheine)
void	holePOInfos(integer aktPO)
void	holePOScheine()
void	holePOs()
void	holeScheintypen()
void	holeScheinvorlagen()
void	holeUebrigeScheine()
void	holeVeranstaltungen()
void	importCoronaVeranstaltungen(String matrikelListe, String aktCorona)
void	importVeranstaltungen()
void	isParam(mixed name, mixed method) isParam() testet auf die Existenz des angegebenen Aufrufparameters Reihenfolge des Ueberpruefung: GET + GET_X, POST + POST_X $name: Name des Parameters $method: Uebertragungsmethode (GET, POST, beides)
void	isPrintableScheinSO(mixed scheintyp, mixed schein, mixed student) isPrintableScheinSO($scheintyp, $schein, $student) $scheintyp: der Typ des Scheins $schein: die ID des Scheins $student: die ID des Studenten returns FALSE wenn keine Scheinvorlage existiert TRUE wenn eine Scheinvorlage existiert
void	isPrintableScheinSO01(mixed schein, mixed student)
void	isPrintableScheinSO02(mixed schein, mixed student)
void	isPrintableScheinSO03(mixed schein, mixed student)
void	isPrintableScheinSO04(mixed schein, mixed student)
void	isPrintableScheinSO05(mixed schein, mixed student)
void	isPrintableScheinSO07(mixed schein, mixed student)
void	isPrintableScheinStudent(mixed scheintyp, mixed schein, mixed studenten, bool scheintypOrigin) isPrintableScheinStudent($scheintyp, $schein, $studenten) $scheintyp: der Typ des Scheins $schein: die ID des Scheins $studenten: Array mit Studenten IDs returns FALSE wenn der Student nicht die Scheinvergabekriterien erfuellt ('status' => SCHEINDRUCK_GESPERRT) wenn keine Scheinvorlage existiert ('status' => SCHEINDRUCK_OK) wenn eine Scheinvorlage existiert und Student in Veranstaltung und nicht in LogScheindruck eingetragen ist ('status' => SCHEINDRUCK_GEDRUCKT, 'zeitpunkt', 'benutzer') wenn Student in Veranstaltung und in LogScheindruck eingetragen ist

void	isPrintableScheinStudent01(mixed schein, mixed studenten, mixed scheintypOrigin)
void	isPrintableScheinStudent02(mixed schein, mixed studenten, mixed scheintypOrigin)
void	isPrintableScheinStudent03(mixed schein, mixed studenten, mixed scheintypOrigin)
void	isPrintableScheinStudent04(mixed schein, mixed studenten, mixed scheintypOrigin)
void	isPrintableScheinStudent05(mixed schein, mixed studenten, mixed scheintypOrigin)
void	isPrintableScheinStudent07(mixed schein, mixed studenten, mixed scheintypOrigin)
void	isPrintableTeilleistung(mixed schein, mixed veranstaltungID, mixed art, mixed studenten, mixed scheintypOrigin)
void	isSemesterBezogen(mixed scheintyp) isSemesterBezogen($scheintyp) $scheintyp: der Typ des Scheins returns TRUE wenn der Scheintyp semesterbezogen ist FALSE wenn der Scheintyp nicht semesterbezogen ist
void	isSemesterBezogen01()
void	isSemesterBezogen02()
void	isSemesterBezogen03()
void	isSemesterBezogen04()
void	isSemesterBezogen05()
void	isSemesterBezogen07()
void	kohorteStudent(String selected_kohorten, String aktVeranstaltung)
int	kumuliertGewichtung(Array teilleistungen, Array berechnungsDetails, String aktTeilleistungName)
int	kumuliertStudienordnung(Array teilleistungen, String aktTeilleistungName)
void	ladeImport()
void	ladePJUpload() ladePJUpload() laedt die Startseite des PJ-Zulassungs-Uploads
void	ladeTabelle()
void	ladeTeilleistungsliste()
void	ladeUpload(mixed aktVeranstaltung) ladeUpload() laedt die Startseite des Noten-Uploads $aktVeranstaltung: Veranstaltung, bei der die Noten eingetragen werden sollen
void	leistungsnachweis(mixed daten, mixed abschnitt, mixed fussnoten, mixed vorschau) leistungsnachweis() erzeugt den endgueltigen Leistungsnachweis $daten: kompletter Datensatz eines Studenten (Struktur: s.
void	leistungsuebersicht(mixed daten, mixed abschnitt, mixed fussnoten, mixed vorschau) leistungsuebersicht() erzeugt die informelle Studenten-Notenuebersicht $daten: kompletter Datensatz eines Studenten (Struktur: s.
void	leistungsuebersichtQuerformat(mixed daten, mixed abschnitt, mixed fussnoten, mixed vorschau) leistungsuebersicht() erzeugt die informelle Studenten-Notenuebersicht $daten: kompletter Datensatz eines Studenten (Struktur: s.
void	letzteSucheAnzeigen()

void	loescheLock(mixed id, mixed art) loescheLock() loescht den Lock auf einen Datensatz $id: die ID des Datensatzes $art: die Art des Locks
void	loeschen()
void	login(mixed origin)
void	logout()
void	noteneintragungRueckgaengig(mixed id, mixed veranstaltungid, mixed studentid, mixed benutzerid, mixed zeitpunkt, mixed aktion, mixed note) eintragen: aus Veranstaltung austragen (wenn noch eingetragen), Eintrag und alle Ereignisse die danach folgen loeschen aendern: auf vorherige Note zuruecksetzen und evtl.
void	notenuebersicht(mixed studenten, mixed abschnitt, mixed art, bool header, bool vorschau, bool print, int start, int limit, int v, bool gruppierung) Bereitet die Daten fuer die Notenuebersicht vor $studenten: Array mit StudentenIDs (Sortierung ist zu beachten!) $abschnitt: Studienabschnitt (VK, K, VK_K), fuer den die Uebersicht erstellt werden soll $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $header: Gibt an, ob ein pdf-Header erstellt werden soll $vorschau: Gibt an fuer Leistungsnachweis/-uebersicht an, ob es sich um eine Druckvorschau handelt $print: TRUE, wenn die Uebersichten direkt gedruckt werden sollen, ansonsten werden die $daten zurueckgegeben (z.B.
void	notenuebersichtQuerformat(mixed studenten, mixed abschnitt, mixed art, bool header, bool vorschau, bool print, int start, int limit, int v, bool gruppierung) Bereitet die Daten fuer die Notenuebersicht im Querformat vor $studenten: Array mit StudentenIDs (Sortierung ist zu beachten!) $abschnitt: Studienabschnitt (VK, K, VK_K), fuer den die Uebersicht erstellt werden soll $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $header: Gibt an, ob ein pdf-Header erstellt werden soll $vorschau: Gibt an fuer Leistungsnachweis/-uebersicht an, ob es sich um eine Druckvorschau handelt $print: TRUE, wenn die Uebersichten direkt gedruckt werden sollen, ansonsten werden die $daten zurueckgegeben (z.B.
void	notenzeile(mixed bezeichnung, mixed kurstyp, mixed stunden, mixed ects, mixed benotung, str notenzusatz, mixed zeilentyp, mixed spaltentyp, str vorname, str name, str matrikel) notenzeile() erstellt eine Zeile der Notenuebersicht fuer einen Schein $bezeichnung: Scheintitel $kurstyp: Kurstyp des Scheins $stunden: Gesamtzahl der Stunden des Scheins $ects: ECTS-Punkte zum Schein $benotung: Note des Scheins $notenzusatz: Noten-Ergaenzung a la 'Inland' (*), 'Ausland' (**),...
void	notenzeileQuerformat(mixed nr, mixed bezeichnung, mixed scheinvorlageTitel, mixed benotung, mixed notenzusatz, mixed datum, mixed semester, mixed zeitraum, mixed scheintyp) notenzeile() erstellt eine Zeile der Notenuebersicht fuer einen Schein $nr: Laufende Nummer des Scheins $bezeichnung: Scheintitel $benotung: Note des Scheins $notenzusatz: Noten-Ergaenzung a la 'Inland' (*), 'Ausland' (**),...
void	notizAnlegen()

void	notizBearbeiten()
void	notizLoeschen()
void	notizSpeichern()
void	pjWahlfachAnlegen(mixed modus)
void	pjWahlfachBearbeiten(mixed id, mixed useDB, mixed modus)
void	pjWahlfaecherAnzeigen()
void	printArray(mixed array, bool exit) printArray() gibt den Inhalt eines Array formatiert aus $array: anzuzeigendes Array $exit: wenn TRUE bricht printArray die Ausfuehrung mit exit ab
void	restorePreviousSeitenmodus()
void	saveSeitenmodus(mixed seitenmodus)
void	scheinAnlegen(mixed modus)
void	scheinBearbeiten(mixed id, mixed useDB, mixed modus)
void	scheindruck(mixed scheine, mixed semester, bool vorschau, bool uebersicht, int student, str teilleistung, str veranstaltung)
void	scheindruckTeilleistungen(mixed scheine, mixed semester, bool vorschau, bool uebersicht, int student, str teilleistung, str veranstaltung)
void	scheineAnzeigen()
void	semesterAnlegen(mixed modus)
void	semesterAnzeigen()
void	semesterBearbeiten(mixed id, mixed useDB, mixed modus)
void	semesterDBErstellen(mixed art, mixed jahr, mixed anfang, mixed ende, mixed vorlesunganfang, mixed vorlesungende)
void	semesterKurz(mixed jahr, mixed art)
void	semesterLang(mixed jahr, mixed art)
void	session()
void	sessionBeenden()
void	setPaneInfo(mixed id) setPaneInfo() holt alle benoetigten Daten des Studenten zur Ansicht in der TabbedPane.
void	setSuche(mixed suche)
void	setzeLock(mixed id, mixed art, mixed benutzer) setzeLock() versucht, einen Lock auf einen Datensatz zu setzen.
void	shownode(mixed x)
void	signaturAngehaengt()
void	signaturHochladen()
void	signaturLoeschen()
void	signaturTest()
void	smartyConstant(mixed key, mixed value) smartyConstant() definiert eine ubergebene Variable als PHP-Konstante und Smarty-Variable $key: Name der Konstante/Smarty-Variable $value: Wert der Konstante/Smarty-Variable
void	smarty_function_button(mixed params, mixed smarty)
void	smarty_function_checkbox(mixed params, mixed smarty)
void	smarty_function_createLink(mixed params, mixed smarty)
void	smarty_function_dropdown(mixed params, mixed smarty)
void	smarty_function_fehler(mixed params, mixed smarty)

void	smarty_function_graphbutton(mixed params, mixed smarty)
void	smarty_function_hinweis(mixed params, mixed smarty)
void	smarty_function_kohortenuebersicht(mixed params, mixed smarty)
void	smarty_function_leistungsuebersicht(mixed params, mixed smarty)
void	smarty_function_loadCustomCss(mixed params, mixed smarty)
void	smarty_function_pouebersicht(mixed params, mixed smarty)
void	smarty_function_radiobutton(mixed params, mixed smarty)
void	smarty_function_submitbutton(mixed params, mixed smarty)
void	smarty_function_textarea(mixed params, mixed smarty)
void	smarty_function_textfield(mixed params, mixed smarty)
void	smarty_function_tor(mixed params, mixed smarty)
void	smarty_function_uebersichtNoteneintragung(mixed params, mixed smarty)
void	smarty_function_uebersichtNotiz(mixed params, mixed smarty)
void	smarty_function_uebersichtTeilNoteneintragung(mixed params, mixed smarty)
void	smarty_function_upload(mixed params, mixed smarty)
string|void	smarty_modifier_date_format(string string, string format, string default_date) Smarty date_format modifier pluginType: modifier Name: date_format Purpose: format datestamps via strftime Input: string: input date string format: strftime format for output - default_date: default date if $string is empty
void	smarty_modifier_escapeJavascript(mixed string)
void	smarty_modifier_stripSlashes(mixed string)
void	smarty_modifier_truncateCenter(mixed string, int length, str etc)
void	speichereFehler(String art, String system, String fehler, integer scheinID, integer studentID, String message, integer ampelfarbe, boolean checked)
void	speichern()
void	standard()
void	startPlausiChecks() Startet die Plausibilitäts-Checks durch einen AJAX-Request.
If	str_split($string (string), $split_length (int), mixed string, int split_length) Convert a string to an array (needed for PHP4 compatibility)
void	studentAnlegen(mixed modus)
void	studentBearbeiten(mixed id, mixed useDB, mixed modus)
void	studentenAnzeigen(mixed suche) studentenAnzeigen($suche) zeigt die Ergebnisse der Studentensuche an $suche: Der Suchstring
void	studentendaten(mixed key, mixed value) studentendaten() erzeugt die zeilenweisen Studentendaten $key: Beschreibung des Datums $value: Datums-Wert
void	studentendetails(mixed studentID, mixed aktSemester) studentendetails() zeigt die Detailansicht (nach Semestern sortierte Scheine + FLNs) eines Studenten an $studentID: anzuzeigender Student $aktSemester: anzuzeigendes Semester
void	toggleNavVisible(mixed id)

void	tor(mixed daten, mixed fussnoten, mixed noten, mixed kurstypen, bool vorschau) tor() erzeugt den transcript of records $daten: kompletter Datensatz eines Studenten (Struktur: s.
void	torAnzeigen(mixed studentid)
void	torFooter() torFooter() erzeugt die Fusszeile des Transcript of Records (Seitenanzahl)
void	torFussnoten(mixed fussnoten, mixed abstand) torFussnoten() erzeugt die zu den Notenuebersichten gehoerenden Fussnoten $fussnoten: Anzuzeigende Fussnoten
void	torHeader(mixed vorname, mixed name, mixed matrikel) torHeader() erzeugt die Kopfzeile des Transcript of Records (Uni-Logo und Student) $vorname: Vorname des Studenten $name: Name des Studenten $matrikel: Matrikelnummer des Studenten
void	uebersicht()
void	uebersichtAnzeigen(mixed studentid, mixed abschnitt)
void	uebersichtK()
void	uebersichtVK()
void	umleiten(str origin)
void	update()
void	updatePO()
void	veranstaltungAnlegen(mixed modus)
void	veranstaltungBearbeiten(mixed id, mixed useDB, mixed modus)
void	veranstaltungenAnlegen()
void	veranstaltungenAnzeigen()
void	zeigeCSVUpload(String aktVeranstaltung:, mixed aktVeranstaltung)
void	zeigeImport() zeigeImport() holt sucht die Studienordnungen heraus und zeigt die Startseite des Studentenimports an
void	zeigeSeite(str fehler)
void	zeitspanneInMonaten(String datumErstePruefung, String datumZweitePruefung)

Function Detail

abbrechen

bs/blauessystem/verw13.php at line 231

```
public void abbrechen()
```

---

aktualisiereFehlerAnzeige

bs/blauessystem/include/plausichecks.inc at line 300

```
public void aktualisiereFehlerAnzeige()
```

Desc:

Aktualisiert die globale Fehleranzeige. Sobald die Seite gewechselt/aktualisiert wird, ist dies aktuell.

---

anschriftzeile

bs/blauessystem/include/pdf/tor.inc at line 36

```
public void anschriftzeile(mixed inhalt, bool emph, bool indent, int width)
```

anschriftzeile() erzeugt eine Zeile der Anschrift $value: Inhalt der Zeile $emph: Fettdruck? $indent: Soll das Feld eingerueckt werden? $width: Breite des Feldes

---

arithmetischesMittel

bs/blauessystem/include/teilleistungen.inc at line 54

```
public int arithmetischesMittel(Array teilleistungen, Array berechnungsDetails,
String aktTeilleistungName)
```

Desc:

Arithmetisches Mittel über die Teilleistungen bilden.

Parameters:

teilleistungen - alle vorhandenen Teilleistungen des Studentens zur Veranstaltung
berechnungsDetails - enthält die Berechnungsdetails
aktTeilleistungName - der Name der aktuell importierten Teilleistung

Returns:

$gesamtNote die berechnete Gesamtnote

---

bearbeiten

bs/blauessystem/verw13.php at line 84

```
public void bearbeiten()
```

---

benutzerAnlegen

bs/blauessystem/sys01.php at line 19

```
public void benutzerAnlegen(mixed modus)
```

---

benutzerAnzeigen

bs/blauessystem/sys02.php at line 22

```
public void benutzerAnzeigen()
```

---

benutzerBearbeiten

bs/blauessystem/sys02.php at line 50

```
public void benutzerBearbeiten(mixed id, mixed useDB, mixed modus)
```

---

berechneGesamtnote

bs/blauessystem/include/teilleistungen.inc at line 18

```
public void berechneGesamtnote($aktTeilleistungName die, mixed aktVeranstaltung,
mixed studentID, mixed aktTeilleistungName)
```

Desc:

Berechnet die Gesamtnote der Teilleistungen

Parameters:

die - aktuelle Veranstaltung
die - Bezeichnung der aktuellen Teilleistung

---

besitztLock

bs/blauessystem/include/general.inc at line 212

```
public void besitztLock(mixed id, mixed art, mixed benutzer)
```

besitztLock() ueberprueft ob der Benutzer noch einen gueltigen Lock auf den Datensatz besitzt $id: die ID des Datensatz $art: die Art des Locks $benutzer: die ID des bearbeitenden Benutzers

---

checkCoronaVeranstaltungen

bs/blauessystem/verw15.php at line 112

```
public void checkCoronaVeranstaltungen()
```

Desc:

Überprüft, ob für das aktuell gewählte Semester schon Veranstaltungen aus Corona vorhanden sind. Ansonsten können die LSF-IDs (MEDXXX) Nummern nicht eingetragen/ausgewählt werden.

---

checkFristKlausurversuche

bs/blauessystem/include/plausichecks.inc at line 18

```
public void checkFristKlausurversuche(Array erg,, String system, boolean
checked, mixed erg)
```

Desc:

Nach Fristüberschreitung der Monatsfrist und der Klausurversuche suchen.

---

checkTeilleistungen

bs/blauessystem/include/plausichecks.inc at line 118

```
public void checkTeilleistungen(Array verg, mixed system, mixed checked)
```

Desc:

Alle Teilleistungen in der DB auf Fristüberschreitungen prüfen.

---

checkVoraussetzungen

bs/blauessystem/include/plausichecks.inc at line 194

```
public void checkVoraussetzungen(String matrikelListe, String aktCorona, mixed
matrikelArray, mixed system, int checked)
```

Desc:

Nach Scheinen für die Veranstaltung suchen. Anschließend prüfen, ob es Scheinkriterien zu diesem Schein gibt ...

---

cmp_scheinDaten

bs/blauessystem/include/scheindruck.inc at line 33

```
public void cmp_scheinDaten(mixed a, mixed b)
```

---

cmp_scheinDatenTL

bs/blauessystem/include/scheindruckTeilleistungen.inc at line 32

```
public void cmp_scheinDatenTL(mixed a, mixed b)
```

---

cmp_scheinansicht

bs/blauessystem/stud02.php at line 48

```
public void cmp_scheinansicht(mixed a, mixed b)
```

Sortiere zunaechst nach Grupperiung (ELN,BP,...), dann nach Sortierung laut SO-Schein-Reihenfolge

---

cmp_scheine

bs/blauessystem/include/scheindruck.inc at line 20

```
public void cmp_scheine(mixed a, mixed b)
```

---

cmp_scheineTL

bs/blauessystem/include/scheindruckTeilleistungen.inc at line 19

```
public void cmp_scheineTL(mixed a, mixed b)
```

---

cmp_suggestions

bs/blauessystem/catsuche.php at line 14

```
public void cmp_suggestions(mixed a, mixed b)
```

---

computeAverage

bs/blauessystem/sys04.php at line 24

```
public void computeAverage(mixed data, int count)
```

---

computeDeviation

bs/blauessystem/sys04.php at line 34

```
public void computeDeviation(mixed data, mixed avg, int count)
```

---

datensatzExistiert

bs/blauessystem/include/general.inc at line 224

```
public void datensatzExistiert(mixed id, mixed art)
```

datensatzExistiert() ueberprueft die Existenz des mittels $art angegebenen Datensatzes $id: ID des zu ueberpruefenden Datensatzes $art: Art des Datensatzes, z.B. 'Dozent', 'Semester',...

---

datensatzGesperrt

bs/blauessystem/include/general.inc at line 193

```
public void datensatzGesperrt(mixed id, mixed art)
```

datensatzGesperrt() prueft, ob ein Datensatz gesperrt ist. Falls dies der Fall ist, wird die BenutzerID des sperrenden Users zurueckgegeben. $id: die ID des Datensatzes $art: die Art des Locks

---

defaults

bs/blauessystem/verw14.php at line 212

```
public void defaults()
```

---

delete

bs/blauessystem/verw15.php at line 130

```
public void delete()
```

Desc:

Aus der Übersichtstabelle können Zuordnungen direkt gelöscht werden.

---

dozentAnlegen

bs/blauessystem/verw08.php at line 43

```
public void dozentAnlegen(mixed modus)
```

---

dozentBearbeiten

bs/blauessystem/include/dozent.inc at line 25

```
public void dozentBearbeiten(mixed id, mixed content, mixed modus)
```

---

dozentenAnzeigen

bs/blauessystem/verw09.php at line 25

```
public void dozentenAnzeigen()
```

---

edit

bs/blauessystem/verw15.php at line 144

```
public void edit()
```

Desc:

Ein Schein wurde zum bearbeiten ausgewählt. Nun werden die Kriterien und alle Corona-Veranstaltungen an das Template übergeben.

---

eintraegeProTag

bs/blauessystem/sys04.php at line 50

```
public void eintraegeProTag(mixed tabelle, mixed zeitpunktBeginn, mixed s_von,
```

```
mixed s_bis, mixed s_avg, mixed s_dev)
```

---

eintragungLoeschen

bs/blauessystem/stud01.php at line 256

```
public void eintragungLoeschen()
```

---

einzelschein04

bs/blauessystem/include/pdf/schein04.inc at line 13

```
public void einzelschein04(mixed daten)
```

---

einzelschein05

bs/blauessystem/include/pdf/schein05.inc at line 13

```
public void einzelschein05(mixed daten)
```

---

einzelschein06

bs/blauessystem/include/pdf/schein06.inc at line 13

```
public void einzelschein06(mixed daten)
```

---

einzelschein07

bs/blauessystem/include/pdf/schein07.inc at line 13

```
public void einzelschein07(mixed daten)
```

---

einzelschein08

bs/blauessystem/include/pdf/schein08.inc at line 13

```
public void einzelschein08(mixed daten)
```

---

einzelschein09

bs/blauessystem/include/pdf/schein09.inc at line 13

```
public void einzelschein09(mixed daten)
```

---

einzelschein10

bs/blauessystem/include/pdf/schein10.inc at line 13

```
public void einzelschein10(mixed daten)
```

---

einzelschein11

bs/blauessystem/include/pdf/schein11.inc at line 13

```
public void einzelschein11(mixed daten)
```

---

einzelschein12

bs/blauessystem/include/pdf/schein12.inc at line 13

```
public void einzelschein12(mixed daten)
```

---

einzelschein14

bs/blauessystem/include/pdf/schein14.inc at line 13

```
public void einzelschein14(mixed daten)
```

---

erstelleScheinNotenlisten

bs/blauessystem/include/notenuebersicht.inc at line 18

```
public void erstelleScheinNotenlisten(mixed studienordnungen, mixed abschnitt,
mixed art, mixed daten, int start, int limit, int v, bool gruppierung)
```

Sucht fuer alle uebergebenen Studienordnungen die entsprechenden Scheinlisten (inkl. Scheintyp) heraus und sucht fuer alle Studenten einer SO die Noten zusammen $studienordnungen: Array mit Studienordnungen (als Key) und zugehoerigen Studenten $abschnitt: Studienabschnitt (VK, K, VK_K), fuer den die Schein-/Notenliste erstellt wird $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $daten: Komplettes (Notenuebersichts-)Datenarray, in welches die Schein-/Notenliste zu den entspr. Studenten

gemergt wird

---

bs/blauessystem/include/notenuebersichtQuerformat.inc at line 18

erstelleScheinNotenlistenQuerformat

```
public void erstelleScheinNotenlistenQuerformat(mixed studienordnungen, mixed
abschnitt, mixed art, mixed daten, int start, int limit, int v, bool
gruppierung)
```

Sucht fuer alle uebergebenen Studienordnungen die entsprechenden Scheinlisten (inkl. Scheintyp) heraus und sucht fuer alle Studenten einer SO die Noten zusammen $studienordnungen: Array mit Studienordnungen (als Key) und zugehoerigen Studenten $abschnitt: Studienabschnitt (VK, K, VK_K), fuer den die Schein-/Notenliste erstellt wird $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $daten: Komplettes (Notenuebersichts-)Datenarray, in welches die Schein-/Notenliste zu den entspr. Studenten gemergt wird

---

erstelleXMLNotenListe

bs/blauessystem/ws.php at line 56

```
public void erstelleXMLNotenListe(String matrikelnummer)
```

Erstellt anhand der Matrikelnummer eine XML Liste, die vom Android ausgelesen wird.

---

fehler

bs/blauessystem/verw13.php at line 255

```
public void fehler()
```

---

formFehler

bs/blauessystem/include/general.inc at line 115

```
public void formFehler(mixed typid, mixed typ, mixed feldid, mixed fehlertext,
mixed previousSeitenmodus)
```

formFehler() zeigt Smarty an, dass es bei einer Formulareingabe einen Fehler gegeben hat $typid: ID des Typs $typ: Typ der Verwaltungsseite $feldid: ID des Formfeldes, das den Fehler enthaelt $fehlertext: anzuzeigender Fehlertext $previousSeitenmodus: wenn TRUE, auf letzten Seitenmodus zuruecksetzen

---

generiereScheine

bs/blauessystem/include/scheindruck.inc at line 218

```
public void generiereScheine(mixed scheinDaten, mixed uebersicht)
```

generiereScheine($scheinDaten, $uebersicht) $scheinDaten enthaelt fuer jede Student/Schein-Kombination folgendes Array: (scheinid, vorlage, vorschau, studentid, matrikel, name, vorname, geburtsdatum, geburtsort, titel, lpa, note, noten, von, bis, datum, halbjahr, dozenten, sig)

---

generiereScheineTL

bs/blauessystem/include/scheindruckTeilleistungen.inc at line 217

```
public void generiereScheineTL(mixed scheinDaten, mixed uebersicht)
```

generiereScheine($scheinDaten, $uebersicht) $scheinDaten enthaelt fuer jede Student/Schein-Kombination folgendes Array: (scheinid, vorlage, vorschau, studentid, matrikel, name, vorname, geburtsdatum, geburtsort, titel, lpa, note, noten, von, bis, datum, halbjahr, dozenten, sig)

---

generiereUebersicht

bs/blauessystem/include/notenuebersicht.inc at line 272

```
public void generiereUebersicht(mixed daten, mixed art, mixed header, mixed
vorschau, bool return)
```

Gibt eine Notenuebersicht des oder der Studenten aus $daten: Array mit allen zu druckenden Daten Aufbau: Studentenweise mit folgenden Feldern: ID, Name, Vorname, Matrikel, Strasse, PLZ, Ort, Geburtsort, Geburtsdatum, Famulatur, Datum, Scheine[VK/K] => Array(Scheintyp, Titel, LPA, Note, Datum) $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $header: Gibt an, ob ein pdf-Header erstellt werden soll $vorschau: Gibt an, ob es sich um eine Druckvorschau handelt $return: Gibt an, wie die PDF-Datei zurueck gegeben werden soll TRUE: Rueckgabe String (fuer Notenspiegel); FALSE: Rueckgabe PDF

---

generiereUebersichtQuerformat

bs/blauessystem/include/notenuebersichtQuerformat.inc at line 298

```
public void generiereUebersichtQuerformat(mixed daten, mixed art, mixed header,
mixed vorschau, bool return)
```

Gibt eine Notenuebersicht des oder der Studenten aus $daten: Array mit allen zu druckenden Daten Aufbau: Studentenweise mit folgenden Feldern: ID, Name, Vorname, Matrikel, Strasse, PLZ, Ort, Geburtsort, Geburtsdatum, Famulatur, Datum, Scheine[VK/K] => Array(Scheintyp, Titel, LPA, Note, Datum) $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $header: Gibt an, ob ein pdf-Header erstellt werden soll $vorschau: Gibt an, ob es sich um eine Druckvorschau handelt $return: Gibt an, wie die PDF-Datei zurueck gegeben werden soll TRUE: Rueckgabe String (fuer Notenspiegel); FALSE: Rueckgabe PDF

---

geschlechtsUmwandlung

bs/blauessystem/verw03.php at line 24

```
public void geschlechtsUmwandlung(mixed geschlecht)
```

geschlechtsUmwandlung() aendert den String 'männlich' bzw. 'weiblich' in 'm' und 'w' um $geschlecht: gibt das Geschlecht an; gueltig: 'männlich' und 'weiblich' gibt bei 'korrektem' Geschlecht 'm' und 'w' zurueck, ansonsten ''

---

getAllDozenten

bs/blauessystem/include/general.inc at line 382

```
public void getAllDozenten()
```

Sucht alle vorhandenen Dozenten aus der Datenbank und liefert diese zurueck

---

getAllDozentenSignaturen

bs/blauessystem/include/general.inc at line 404

```
public void getAllDozentenSignaturen(mixed dozenten)
```

Durchsucht das Signatur-Verzeichnis nach Unterschriften der uebergebenen Dozenten und schreibt das Ergebnis (Signatur vorhanden ja/nein) in das Rueckgabe-Array

---

getAllScheine

bs/blauessystem/include/general.inc at line 349

```
public void getAllScheine()
```

Sucht alle vorhandenen Scheine aus der Datenbank und gruppiert diese nach VK/K/WF.

---

getAllSemester

bs/blauessystem/include/general.inc at line 326

```
public void getAllSemester(bool auchOhneVeranstaltungen)
```

Sucht alle vorhandenen Semester mit zugehoeriger Anzahl von Veranstaltungen aus der Datenbank.

---

getArt

bs/blauessystem/include/session.inc at line 99

```
public void getArt()
```

---

getBenutzer

bs/blauessystem/include/session.inc at line 76

```
public void getBenutzer()
```

---

getBenutzerName

bs/blauessystem/include/session.inc at line 82

```
public void getBenutzerName()
```

---

getBesteVeranstaltung

bs/blauessystem/include/scheintyp.inc at line 528

```
public void getBesteVeranstaltung(mixed scheintyp, mixed schein, mixed
studenten)
```

Liefert die beste Veranstaltung fuer jeden Studenten zurueck, die der Student fuer diesen Schein besucht hat.

---

getBesteVeranstaltung01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 436

```
public void getBesteVeranstaltung01(mixed schein, mixed studenten, bool
zweites_wahlfach)
```

---

getBesteVeranstaltung02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 465

```
public void getBesteVeranstaltung02(mixed schein, mixed studenten)
```

---

getBesteVeranstaltung03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 199

```
public void getBesteVeranstaltung03(mixed schein, mixed studenten)
```

---

getBesteVeranstaltung04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 140

```
public void getBesteVeranstaltung04(mixed schein, mixed studenten, bool
zweites_wahlfach)
```

---

getBesteVeranstaltung05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 371

```
public void getBesteVeranstaltung05(mixed schein, mixed studenten)
```

---

getBesteVeranstaltung07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 146

```
public void getBesteVeranstaltung07(mixed schein, mixed studenten)
```

---

getCounterScheinSemester

bs/blauessystem/include/scheintyp.inc at line 439

```
public void getCounterScheinSemester(mixed scheintyp, mixed schein, mixed
semester, bool scheintypOrigin)
```

getCounterScheinSemester($scheintyp, $schein, $semester) $scheintyp: der Typ des Scheins $schein: die ID des Scheins $semester: die ID des Semesters returns Array (TRUE, gedruckt, insgesamt) wenn Studenten in Veranstaltungen und nicht in LogScheindruck eingetragen sind

Array (FALSE, gedruckt, insgesamt) wenn Studenten in Veranstaltungen und in LogScheindruck eingetragen sind

---

getCounterScheinSemester01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 362

```
public void getCounterScheinSemester01(mixed schein, mixed semester, mixed scheintypOrigin)
```

---

getCounterScheinSemester02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 367

```
public void getCounterScheinSemester02(mixed schein, mixed semester, mixed scheintypOrigin)
```

---

getCounterScheinSemester03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 129

```
public void getCounterScheinSemester03(mixed schein, mixed semester, mixed scheintypOrigin)
```

---

getCounterScheinSemester04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 122

```
public void getCounterScheinSemester04(mixed schein, mixed semester, mixed scheintypOrigin)
```

---

getCounterScheinSemester05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 298

```
public void getCounterScheinSemester05(mixed schein, mixed semester, mixed scheintypOrigin)
```

---

getCounterScheinSemester07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 128

```
public void getCounterScheinSemester07(mixed schein, mixed semester, mixed scheintypOrigin)
```

---

getDatum

bs/blauessystem/include/scheintyp.inc at line 191

```
public void getDatum(mixed scheintyp, mixed schein, mixed studenten, mixed typ)
```

Liefert das Datum (von, bis, datum) des Scheins fuer jeden uebergebenen Studenten zurueck.

---

getDatum01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 168

```
public void getDatum01(mixed schein, mixed studenten, mixed typ)
```

---

getDatum02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 130

```
public void getDatum02(mixed schein, mixed studenten, mixed typ)
```

---

getDatum03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 57

```
public void getDatum03(mixed schein, mixed studenten, mixed typ)
```

---

getDatum04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 95

```
public void getDatum04(mixed schein, mixed studenten, mixed typ)
```

---

getDatum05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 120

```
public void getDatum05(mixed schein, mixed studenten, mixed typ)
```

getDatum07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 101

```
public void getDatum07(mixed schein, mixed studenten, mixed typ)
```

getDays

bs/blauessystem/sys04.php at line 17

```
public void getDays(mixed date1, mixed date2)
```

getDetails

bs/blauessystem/include/scheintyp.inc at line 21

```
public void getDetails(mixed scheintyp, mixed schein, mixed studenten)
```

Liefert die Details (Titel, LPA-Code, Note, Untertitel, Datum, Halbjahr, Dozenten, Signaturen) zurueck. Die Methode erwartet, dass $studenten die Scheinkriterien erfuellen und ueberprueft diese nicht nochmals

getDetails01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 10

```
public void getDetails01(mixed schein, mixed studenten, bool zweites_wahlfach)
```

getDetails02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 10

```
public void getDetails02(mixed schein, mixed studenten)
```

getDetails03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 10

```
public void getDetails03(mixed schein, mixed studenten)
```

getDetails04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 10

```
public void getDetails04(mixed schein, mixed studenten, bool zweites_wahlfach)
```

getDetails05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 10

```
public void getDetails05(mixed schein, mixed studenten)
```

getDetails07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 10

```
public void getDetails07(mixed schein, mixed studenten)
```

getDistinctUnterscheine02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 485

```
public void getDistinctUnterscheine02(mixed schein)
```

getDozent

bs/blauessystem/include/scheintyp.inc at line 135

```
public void getDozent(mixed scheintyp, mixed schein, mixed studenten)
```

Liefert die Dozenten des Scheins fuer jeden uebergebenen Studenten zurueck.

getDozent01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 102

```
public void getDozent01(mixed schein, mixed studenten)
```

getDozent02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 76

```
public void getDozent02(mixed schein, mixed studenten)
```

getDozent03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 49

```
public void getDozent03(mixed schein, mixed studenten)
```

getDozent04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 87

```
public void getDozent04(mixed schein, mixed studenten)
```

getDozent05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 66

```
public void getDozent05(mixed schein, mixed studenten)
```

getDozent07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 49

```
public void getDozent07(mixed schein, mixed studenten)
```

getHalbjahr

bs/blauessystem/include/scheintyp.inc at line 222

```
public void getHalbjahr(mixed scheintyp, mixed schein, mixed studenten)
```

Liefert die Werte 1, 2 oder 3 zurueck, mit denen die Halbjahre angezeigt werden, in denen der Student die Veranstaltungen fuer diesen Schein besucht hat. (1 => Sommersemester, 2 => Wintersemester, 3 => Sommer- und Wintersemester)

getHalbjahr01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 194

```
public void getHalbjahr01(mixed schein, mixed studenten)
```

getHalbjahr02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 182

```
public void getHalbjahr02(mixed schein, mixed studenten)
```

getHalbjahr03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 61

```
public void getHalbjahr03(mixed schein, mixed studenten)
```

getHalbjahr04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 99

```
public void getHalbjahr04(mixed schein, mixed studenten)
```

getHalbjahr05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 145

```
public void getHalbjahr05(mixed schein, mixed studenten)
```

getHalbjahr07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 105

```
public void getHalbjahr07(mixed schein, mixed studenten)
```

getJahr

bs/blauessystem/include/session.inc at line 95

```
public void getJahr()
```

getKohortenAnzahlViaVeranstaltung

bs/blauessystem/schein05.php at line 334

```
public void getKohortenAnzahlViaVeranstaltung(String aktVeranstaltung)
```

Desc:

Gibt die Kohorten und die jeweilige Anzahl der Prüflinge zurück, die in dieser Veranstaltung die Prüfung abgelegt haben.

---

getLPA

bs/blauessystem/include/scheintyp.inc at line 107

```
public void getLPA(mixed scheintyp, mixed schein)
```

Liefert den LPA-Code des Scheins zurueck.

---

getLPA01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 93

```
public void getLPA01(mixed schein)
```

---

getLPA02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 72

```
public void getLPA02(mixed schein)
```

---

getLPA03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 45

```
public void getLPA03(mixed schein)
```

---

getLPA04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 83

```
public void getLPA04(mixed schein)
```

---

getLPA05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 62

```
public void getLPA05(mixed schein)
```

---

getLPA07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 45

```
public void getLPA07(mixed schein)
```

---

getLeistungsuebersichtdaten

bs/blauessystem/notenuebersicht.php at line 15

```
public void getLeistungsuebersichtdaten(mixed abschnitt, mixed kohorten, mixed
sortierung)
```

---

getNavVisible

bs/blauessystem/include/session.inc at line 177

```
public void getNavVisible()
```

---

getNavigation

bs/blauessystem/include/session.inc at line 156

```
public void getNavigation()
```

---

getNote

bs/blauessystem/include/scheintyp.inc at line 252

```
public void getNote(mixed scheintyp, mixed schein, mixed studenten, mixed typ)
```

Liefert die Note des Studenten fuer diesen Schein zurueck. $typ: Gibt das Format der Note an, zulaessige Werte sind NOTE_ID, NOTE_TEXTNOTE, NOTE_TEXT, NOTE_NOTE, NOTE_TEXTINTL.

---

getNote01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 220

```
public void getNote01(mixed schein, mixed studenten, mixed typ, bool
```

```
zweites_wahlfach)
```

---

getNote02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 226

```
public void getNote02(mixed schein, mixed studenten, mixed typ)
```

---

getNote03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 65

```
public void getNote03(mixed schein, mixed studenten, mixed typ)
```

---

getNote04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 103

```
public void getNote04(mixed schein, mixed studenten, mixed typ, bool
zweites_wahlfach)
```

---

getNote05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 171

```
public void getNote05(mixed schein, mixed studenten, mixed typ)
```

---

getNote07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 109

```
public void getNote07(mixed schein, mixed studenten, mixed typ)
```

---

getNotenzusatz

bs/blauessystem/include/scheintyp.inc at line 283

```
public void getNotenzusatz(mixed scheintyp, mixed schein, mixed studenten, mixed
typ)
```

> Liefert den Notenzusatz, falls vorhanden, des Studenten fuer diesen Schein zurueck. $typ: Gibt das Format des Notenzusatzes an, zulaessige Werte sind NOTENZUSATZ_KURZ, NOTENZUSATZ_BESCHREIBUNG, NOTENZUSATZ_BESCHREIBUNGINTL, NOTENZUSATZ_ABKUERZUNG, NOTENZUSATZ_ZEICHEN, NOTENZUSATZ_SCHEINDRUCKBAR

---

getNotenzusatz01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 257

```
public void getNotenzusatz01(mixed schein, mixed studenten, mixed typ)
```

---

getNotenzusatz02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 303

```
public void getNotenzusatz02(mixed schein, mixed studenten, mixed typ)
```

---

getNotenzusatz03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 69

```
public void getNotenzusatz03(mixed schein, mixed studenten, mixed typ)
```

---

getNotenzusatz04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 107

```
public void getNotenzusatz04(mixed schein, mixed studenten, mixed typ)
```

---

getNotenzusatz05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 238

```
public void getNotenzusatz05(mixed schein, mixed studenten, mixed typ)
```

---

getNotenzusatz07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 113

```
public void getNotenzusatz07(mixed schein, mixed studenten, mixed typ)
```

getNotizArten

bs/blauessystem/stud01.php at line 330

```
public void getNotizArten()
```

getPJWahlfach

bs/blauessystem/include/general.inc at line 308

```
public void getPJWahlfach()
```

Liefert die vorhandenen PJ-Wahlfaecher fuer die Studenten-Verwaltungsseiten zurueck (inkl. dem NULL-Wert "kein PJ-Wahlfach ausgewaehlt"

getPOName

bs/blauessystem/note06.php at line 167

```
public void getPOName(integer aktPO)
```

Desc:

Gibt den Titel der Prüfungsordnung zurück.

getParam

bs/blauessystem/include/general.inc at line 61

```
public void getParam(mixed name, mixed method)
```

getParam() gibt einen Aufrufparameter des Skriptes zurueck Reihenfolge des Ueberpruefung: GET, POST $name: Name des Parameters $method: Uebertragungsmethode (GET, POST, beides)

getPostSeitenmodus

bs/blauessystem/include/general.inc at line 78

```
public void getPostSeitenmodus()
```

getPostSeitenmodus() ueberprueft, ob ein Parameter der Form 'seitenmodus_xx' uebergeben wurde, und gibt den Teil 'xx' zurueck. Sollte nur intern verwendet werden!

getPreviousSeitenmodus

bs/blauessystem/include/session.inc at line 146

```
public void getPreviousSeitenmodus()
```

getSO

bs/blauessystem/include/general.inc at line 234

```
public void getSO()
```

getSO() sucht alle vorhandenen Studienordnungen, gliedert sie in VK und K und gibt diese beiden Arrays zurueck

getScheinDetails

bs/blauessystem/include/scheindruck.inc at line 116

```
public void getScheinDetails(mixed schein, mixed studenten)
```

getScheinDetailsTL

bs/blauessystem/include/scheindruckTeilleistungen.inc at line 115

```
public void getScheinDetailsTL(mixed schein, mixed studenten)
```

getScheinSemester

bs/blauessystem/include/scheintyp.inc at line 584

```
public void getScheinSemester(mixed scheintyp, mixed schein, mixed studenten)
```

Einfach nur das Semester zurückliefern.

getScheinVollstaendigkeit

bs/blauessystem/include/general.inc at line 258

```
public void getScheinVollstaendigkeit(mixed studentid, mixed abschnitt)
```

getScheine

bs/blauessystem/include/scheindruck.inc at line 52

```
public void getScheine(mixed semester)
```

getScheineBestanden

bs/blauessystem/include/general.inc at line 274

```
public void getScheineBestanden(mixed studentid, mixed scheine)
```

getScheineSO

bs/blauessystem/include/general.inc at line 290

```
public void getScheineSO(mixed studentid, mixed type)
```

getScheineTL

bs/blauessystem/include/scheindruckTeilleistungen.inc at line 51

```
public void getScheineTL(mixed semester)
```

getSeitenmodus

bs/blauessystem/include/general.inc at line 94

```
public void getSeitenmodus()
```

getSeitenmodus() gibt den uebergebenen Seitenmodus zurueck, oder SEITENMODUS_DEFAULT falls keiner gesetzt wurde. Es werden zuerst POST- und dann GET-Parameter ausgewertet. Zusaetzlich wird der Seitenmodus in der Session abgespeichert und dem Template uebergeben.

getSemester

bs/blauessystem/include/session.inc at line 103

```
public void getSemester()
```

getSignatur

bs/blauessystem/include/scheintyp.inc at line 163

```
public void getSignatur(mixed scheintyp, mixed schein, mixed studenten)
```

Liefert die Dateinamen der Signaturen fuer jeden uebergebenen Studenten zurueck.

getSignatur01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 135

```
public void getSignatur01(mixed schein, mixed studenten)
```

getSignatur02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 103

```
public void getSignatur02(mixed schein, mixed studenten)
```

getSignatur03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 53

```
public void getSignatur03(mixed schein, mixed studenten)
```

getSignatur04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 91

```
public void getSignatur04(mixed schein, mixed studenten)
```

getSignatur05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 94

```
public void getSignatur05(mixed schein, mixed studenten)
```

---

getSignatur07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 75

```
public void getSignatur07(mixed schein, mixed studenten)
```

---

getStudentDetails

bs/blauessystem/include/scheindruck.inc at line 98

```
public void getStudentDetails(mixed studenten)
```

---

getStudentDetailsTL

bs/blauessystem/include/scheindruckTeilleistungen.inc at line 97

```
public void getStudentDetailsTL(mixed studenten)
```

---

getStudentIDs

bs/blauessystem/include/scheintyp.inc at line 499

```
public void getStudentIDs(mixed scheintyp, mixed schein, mixed semester, mixed
logscheindruck, bool studenten, bool scheintypOrigin)
```

---

getStudentIDs01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 385

```
public void getStudentIDs01(mixed schein, mixed semester, mixed logscheindruck,
bool studenten, mixed scheintypOrigin)
```

---

getStudentIDs02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 387

```
public void getStudentIDs02(mixed schein, mixed semester, mixed logscheindruck,
bool studenten, mixed scheintypOrigin)
```

---

getStudentIDs03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 149

```
public void getStudentIDs03(mixed schein, mixed semester, mixed logscheindruck,
bool studenten, mixed scheintypOrigin)
```

---

getStudentIDs04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 133

```
public void getStudentIDs04(mixed schein, mixed semester, mixed logscheindruck,
bool studenten, mixed scheintypOrigin)
```

---

getStudentIDs05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 319

```
public void getStudentIDs05(mixed schein, mixed semester, mixed logscheindruck,
bool studenten, mixed scheintypOrigin)
```

---

getStudentIDs07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 139

```
public void getStudentIDs07(mixed schein, mixed semester, mixed logscheindruck,
bool studenten, mixed scheintypOrigin)
```

---

getStudentenNotenViaKohorten

bs/blauessystem/schein05.php at line 304

```
public void getStudentenNotenViaKohorten(String selected_kohorten, mixed
aktVeranstaltung)
```

Desc:

Ausnahmslos alle Studenten der übergegeben Kohorten und dessen Noten (wenn vorhanden) selektieren.

Parameters:

selected_kohorten - Kommaseparierte Liste von Kohorten-IDs

---

getStudentenNotenViaKohortenOhne

bs/blauessystem/schein05.php at line 262

```
public void getStudentenNotenViaKohortenOhne(String selected_kohorten, mixed
aktVeranstaltung)
```

Desc:

Alle Studenten selektieren, die bereits früher einmal an einer Veranstaltung mit demselben Schein teilgenommen haben.

Parameters:

selected_kohorten - Kommaseparierte Liste von Kohorten-IDs

---

getSuche

bs/blauessystem/include/session.inc at line 127

```
public void getSuche()
```

---

getTitel

bs/blauessystem/include/scheintyp.inc at line 50

```
public void getTitel(mixed scheintyp, mixed schein, mixed studenten, mixed typ)
```

Liefert den Titel des Scheins fuer jeden uebergebenen Studenten zurueck. $typ: Gibt das Format der Note an, zulaessige Werte sind TITEL_DE, TITEL_INTL

---

getTitel01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 63

```
public void getTitel01(mixed schein, mixed studenten, mixed typ, str
teilleistung)
```

---

getTitel02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 37

```
public void getTitel02(mixed schein, mixed studenten, mixed typ)
```

---

getTitel03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 37

```
public void getTitel03(mixed schein, mixed studenten, mixed typ)
```

---

getTitel04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 56

```
public void getTitel04(mixed schein, mixed studenten, mixed typ, bool
zweites_wahlfach)
```

---

getTitel05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 37

```
public void getTitel05(mixed schein, mixed studenten, mixed typ)
```

---

getTitel07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 37

```
public void getTitel07(mixed schein, mixed studenten, mixed typ)
```

---

getUnterscheine

bs/blauessystem/include/scheintyp.inc at line 556

```
public void getUnterscheine(mixed scheintyp, mixed schein)
```

Liefert die Unterscheine (falls vorhanden) fuer diese Schein-/Scheintyp-Kombination zurueck

getUnterscheine01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 462

```
public void getUnterscheine01(mixed schein)
```

getUnterscheine02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 467

```
public void getUnterscheine02(mixed schein)
```

getUnterscheine03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 203

```
public void getUnterscheine03(mixed schein)
```

getUnterscheine04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 144

```
public void getUnterscheine04(mixed schein)
```

getUnterscheine05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 392

```
public void getUnterscheine05(mixed schein)
```

getUnterscheine07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 148

```
public void getUnterscheine07(mixed schein)
```

getUntertitel

bs/blauessystem/include/scheintyp.inc at line 79

```
public void getUntertitel(mixed scheintyp, mixed schein, mixed studenten)
```

Liefert die Untertitel des Scheins fuer jeden uebergebenen Studenten zurueck (z.B. die Unterscheinauflistung mitsamt Noten beim FLN).

getUntertitel01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 89

```
public void getUntertitel01(mixed schein, mixed studenten)
```

getUntertitel02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 41

```
public void getUntertitel02(mixed schein, mixed studenten)
```

getUntertitel03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 41

```
public void getUntertitel03(mixed schein, mixed studenten)
```

getUntertitel04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 79

```
public void getUntertitel04(mixed schein, mixed studenten)
```

getUntertitel05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 41

```
public void getUntertitel05(mixed schein, mixed studenten)
```

getUntertitel07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 41

```
public void getUntertitel07(mixed schein, mixed studenten)
```

getVeranstaltungTitel

bs/blauessystem/schein05.php at line 353

```
public void getVeranstaltungTitel(integer aktVeranstaltung)
```

Desc:

Den Titel der ausgewählten Veranstaltung zurückgeben.

---

getVeranstaltungen05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 373

```
public void getVeranstaltungen05(mixed schein, mixed studenten)
```

---

getVeranstaltungsListe

bs/blauessystem/schein05.php at line 368

```
public void getVeranstaltungsListe()
```

Desc:

Gibt eine Liste der Veranstaltungen zurück, die in diesem Semester angeboten werden.

---

getVersion

bs/blauessystem/include/smarty.inc at line 39

```
public void getVersion()
```

getVersion() gibt die Versionsnummer zurueck

---

globalConfig

bs/blauessystem/stats/stats.php at line 130

```
public void globalConfig(mixed title, mixed vlabel)
```

---

gueltigeSession

bs/blauessystem/include/session.inc at line 47

```
public void gueltigeSession()
```

Prueft ob in der aktuellen Session eine BenutzerID abgelegt ist und diese in der Datenbank noch existiert

---

hasChild

bs/blauessystem/verw01.php at line 219

```
public void hasChild(mixed p)
```

---

hasParams

bs/blauessystem/include/general.inc at line 43

```
public void hasParams(mixed name, mixed quantifier, mixed method)
```

hasParams() testet auf die Existenz aller Aufrufparameter / mindestens eines Aufrufparameters $name: Array mit den Namen der Parameter $quantifier: ANY/ALL Ueberpruefung auf einen/alle Parameter $method: Uebertragungsmethode (GET, POST, beides)

---

headerNotenliste

bs/blauessystem/include/pdf/tor.inc at line 144

```
public void headerNotenliste()
```

headerNotenliste() ist ein Wrapper zum Erstellen der Kopfzeile einer Notenliste

---

hideFehler

bs/blauessystem/verw00.php at line 312

```
public void hideFehler(integer errorid, String checkBS)
```

Desc:

Verbirgt Fehler, bekommt die Fehler-ID und die Begründung übergeben.

---

holeAlleSemester

bs/blauessystem/verw01.php at line 191

```
public void holeAlleSemester()
```

---

holeKlausurversucheFrist

bs/blauessystem/verw14.php at line 41

```
public void holeKlausurversucheFrist(mixed po)
```

---

holeKriterien

bs/blauessystem/verw15.php at line 88

```
public void holeKriterien(Array scheine)
```

Desc:

Lädt zu jedem Schein die passenden ImportKritieren/Scheinzuordnungen. Diese werden dann in der Übersichtstabelle angezeigt.

---

holePOInfos

bs/blauessystem/note06.php at line 153

```
public void holePOInfos(integer aktPO)
```

Desc:

Eine Prüfungsordnung soll kopiert werden. Um das kopieren zu erleichtern, wird eine Übersicht der alten PO aus der DB geladen. Diese kann dann von Hand angepasst werden.

---

holePOScheine

bs/blauessystem/note06.php at line 251

```
public void holePOScheine()
```

Desc:

Hier werden alle Scheine aus der DB geladen, die der Prüfungsordnung zugeordnet sind. Diese werden in der rechten Auswahlbox angezeigt.

---

holePOs

bs/blauessystem/note06.php at line 179

```
public void holePOs()
```

Desc:

Standardfall. Alle Prüfungsordnungen werden in die obere Auswahlliste geladen.

---

holeScheintypen

bs/blauessystem/note06.php at line 304

```
public void holeScheintypen()
```

Desc:

Beim bearbeiten der PO wird eine Auswahlliste der Scheintypen angezeigt. Diese Typen werden hier aus der DB geholt.

---

holeScheinvorlagen

bs/blauessystem/note06.php at line 281

```
public void holeScheinvorlagen()
```

Desc:

Beim bearbeiten der PO wird eine Auswahlliste der Scheinvorlagen angezeigt. Diese

Vorlagen werden hier aus der DB geholt.

---

holeUebrigeScheine

bs/blauessystem/note06.php at line 209

```
public void holeUebrigeScheine()
```

Desc:

Holt alle Scheine aus der Datenbank, die der Prüfungsordnung noch nicht zugeordnet worden sind. Hierfür werden alle Scheine nach dem Typ (Vorklinik/Klinik) gefiltert. Diese werden in der linken Auswahlbox angezeigt.

---

holeVeranstaltungen

bs/blauessystem/verw15.php at line 201

```
public void holeVeranstaltungen()
```

Desc:

Gibt alle Titel und LSFID IDs der aus Corona importierten Veranstaltungen zurück.

---

importCoronaVeranstaltungen

bs/blauessystem/include/corona.inc at line 13

```
public void importCoronaVeranstaltungen(String matrikelListe, String aktCorona)
```

Desc:

Import Veranstaltungen des aktuellen Semesters aus Corona und speichert sie in der Tabelle VeranstaltungCorona

---

importVeranstaltungen

bs/blauessystem/include/importVeranstaltungen.inc at line 12

```
public void importVeranstaltungen()
```

Desc:

Die Corona-Veranstaltungen sollen neu geladen, bzw. aktualisiert werden. Anhand dieser Veranstaltungen kann die Zuordnung der LSF IDs (MEDXXX) stattfinden.

---

isParam

bs/blauessystem/include/general.inc at line 26

```
public void isParam(mixed name, mixed method)
```

isParam() testet auf die Existenz des angegebenen Aufrufparameters Reihenfolge des Ueberpruefung: GET + GET_X, POST + POST_X $name: Name des Parameters $method: Uebertragungsmethode (GET, POST, beides)

---

isPrintableScheinSO

bs/blauessystem/include/scheintyp.inc at line 316

```
public void isPrintableScheinSO(mixed scheintyp, mixed schein, mixed student)
```

isPrintableScheinSO($scheintyp, $schein, $student) $scheintyp: der Typ des Scheins $schein: die ID des Scheins $student: die ID des Studenten returns FALSE wenn keine Scheinvorlage existiert TRUE wenn eine Scheinvorlage existiert

---

isPrintableScheinSO01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 296

```
public void isPrintableScheinSO01(mixed schein, mixed student)
```

---

isPrintableScheinSO02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 311

```
public void isPrintableScheinSO02(mixed schein, mixed student)
```

---

isPrintableScheinSO03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 73

```
public void isPrintableScheinSO03(mixed schein, mixed student)
```

---

isPrintableScheinSO04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 111

```
public void isPrintableScheinSO04(mixed schein, mixed student)
```

---

isPrintableScheinSO05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 242

```
public void isPrintableScheinSO05(mixed schein, mixed student)
```

---

isPrintableScheinSO07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 117

```
public void isPrintableScheinSO07(mixed schein, mixed student)
```

---

isPrintableScheinStudent

bs/blauessystem/include/scheintyp.inc at line 351

```
public void isPrintableScheinStudent(mixed scheintyp, mixed schein, mixed
studenten, bool scheintypOrigin)
```

> isPrintableScheinStudent($scheintyp, $schein, $studenten) $scheintyp: der Typ des Scheins $schein: die ID des Scheins $studenten: Array mit Studenten IDs returns FALSE wenn der Student nicht die Scheinvergabekriterien erfuellt ('status' => SCHEINDRUCK_GESPERRT) wenn keine Scheinvorlage existiert ('status' => SCHEINDRUCK_OK) wenn eine Scheinvorlage existiert und Student in Veranstaltung und nicht in LogScheindruck eingetragen ist ('status' => SCHEINDRUCK_GEDRUCKT, 'zeitpunkt', 'benutzer') wenn Student in Veranstaltung und in LogScheindruck eingetragen ist

---

isPrintableScheinStudent01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 308

```
public void isPrintableScheinStudent01(mixed schein, mixed studenten, mixed
scheintypOrigin)
```

---

isPrintableScheinStudent02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 315

```
public void isPrintableScheinStudent02(mixed schein, mixed studenten, mixed
scheintypOrigin)
```

---

isPrintableScheinStudent03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 77

```
public void isPrintableScheinStudent03(mixed schein, mixed studenten, mixed
scheintypOrigin)
```

---

isPrintableScheinStudent04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 115

```
public void isPrintableScheinStudent04(mixed schein, mixed studenten, mixed
scheintypOrigin)
```

---

isPrintableScheinStudent05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 246

```
public void isPrintableScheinStudent05(mixed schein, mixed studenten, mixed
scheintypOrigin)
```

---

isPrintableScheinStudent07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 121

```
public void isPrintableScheinStudent07(mixed schein, mixed studenten, mixed scheintypOrigin)
```

---

isPrintableTeilleistung

bs/blauessystem/include/scheintyp.inc at line 376

```
public void isPrintableTeilleistung(mixed schein, mixed veranstaltungID, mixed art, mixed studenten, mixed scheintypOrigin)
```

---

isSemesterBezogen

bs/blauessystem/include/scheintyp.inc at line 470

```
public void isSemesterBezogen(mixed scheintyp)
```

isSemesterBezogen($scheintyp) $scheintyp: der Typ des Scheins returns TRUE wenn der Scheintyp semesterbezogen ist FALSE wenn der Scheintyp nicht semesterbezogen ist

---

isSemesterBezogen01

bs/blauessystem/include/scheintyp/scheintyp01.inc at line 381

```
public void isSemesterBezogen01()
```

---

isSemesterBezogen02

bs/blauessystem/include/scheintyp/scheintyp02.inc at line 383

```
public void isSemesterBezogen02()
```

---

isSemesterBezogen03

bs/blauessystem/include/scheintyp/scheintyp03.inc at line 145

```
public void isSemesterBezogen03()
```

---

isSemesterBezogen04

bs/blauessystem/include/scheintyp/scheintyp04.inc at line 129

```
public void isSemesterBezogen04()
```

---

isSemesterBezogen05

bs/blauessystem/include/scheintyp/scheintyp05.inc at line 315

```
public void isSemesterBezogen05()
```

---

isSemesterBezogen07

bs/blauessystem/include/scheintyp/scheintyp07.inc at line 135

```
public void isSemesterBezogen07()
```

---

kohorteStudent

bs/blauessystem/schein05.php at line 178

```
public void kohorteStudent(String selected_kohorten, String aktVeranstaltung)
```

Desc:

Zu den ausgewählten Kohorten und der ausgewählten Veranstaltung alle Studenten aus DB holen, die a) an dieser Veranstaltung teilgenommen haben b) an einer anderen Veranstaltung teilgenommen haben, die demselben Schein zugeordnet ist c) alle restlichen Studenten der Kohorten ausgeben, die noch nie teilgenommen haben.

---

kumuliertGewichtung

bs/blauessystem/include/teilleistungen.inc at line 140

```
public int kumuliertGewichtung(Array teilleistungen, Array berechnungsDetails, String aktTeilleistungName)
```

Desc:

Kumulierte Gewichtung durchführen

Parameters:

teilleistungen - alle vorhandenen Teilleistungen des Studentens zur Veranstaltung
berechnungsDetails - enthält die Berechnungsdetails
aktTeilleistungName - der Name der aktuell importierten Teilleistung

Returns:

$gesamtNote die gewichtete Gesamtnote

---

kumuliertStudienordnung

bs/blauessystem/include/teilleistungen.inc at line 106

```
public int kumuliertStudienordnung(Array teilleistungen, String
aktTeilleistungName)
```

Desc:

Kumuliert die Gesamtzahl der Teilleistungen.

Parameters:

teilleistungen - alle vorhandenen Teilleistungen des Studentens zur Veranstaltung
aktTeilleistungName - der Name der aktuell importierten Teilleistung

Returns:

$gesamtNote die kumulierte Gesamtnote

---

ladeImport

bs/blauessystem/note04.php at line 43

```
public void ladeImport()
```

Desc:

Trägt die importierten Daten in die Datenbank ein.

---

ladePJUpload

bs/blauessystem/pj03.php at line 21

```
public void ladePJUpload()
```

ladePJUpload() laedt die Startseite des PJ-Zulassungs-Uploads

---

ladeTabelle

bs/blauessystem/verw14.php at line 91

```
public void ladeTabelle()
```

---

ladeTeilleistungsliste

bs/blauessystem/note04.php at line 349

```
public void ladeTeilleistungsliste()
```

Desc:

Die Veranstaltung, für die Teilleistungen importiert werden sollen, wurde ausgewählt. Zudem wurde auch die zu importierende Teilleistung ausgewählt. Nun wird der eigentliche Notenimport angezeigt.

---

ladeUpload

bs/blauessystem/note01.php at line 54

```
public void ladeUpload(mixed aktVeranstaltung)
```

ladeUpload() laedt die Startseite des Noten-Uploads $aktVeranstaltung: Veranstaltung, bei der die Noten eingetragen werden sollen

---

leistungsnachweis

bs/blauessystem/include/pdf/leistungsnachweis.inc at line 113

```
public void leistungsnachweis(mixed
daten, mixed abschnitt, mixed fussnoten, mixed vorschau)
```

leistungsnachweis() erzeugt den endgueltigen Leistungsnachweis $daten: kompletter Datensatz eines Studenten (Struktur: s. generiereUebersicht() in notenuebersicht.inc) $abschnitt: auszuwertender Studienabschnitt (VK, K) $fussnoten: anzuzeigende Fussnoten $vorschau: gibt an, ob Wasserzeichen und !Unterschrift auf Seite gedruckt werden soll

---

leistungsuebersicht

bs/blauessystem/include/pdf/leistungsuebersicht.inc at line 115

```
public void leistungsuebersicht(mixed daten, mixed abschnitt, mixed fussnoten,
mixed vorschau)
```

leistungsuebersicht() erzeugt die informelle Studenten-Notenuebersicht $daten: kompletter Datensatz eines Studenten (Struktur: s. generiereUebersicht() in notenuebersicht.inc) $abschnitt: auszuwertender Studienabschnitt (VK, K) $fussnoten: anzuzeigende Fussnoten $vorschau: gibt an, ob Wasserzeichen und !Unterschrift auf Seite gedruckt werden soll

---

bs/blauessystem/include/pdf/leistungsuebersichtQuerformat.inc at line 131

leistungsuebersichtQuerformat

```
public void leistungsuebersichtQuerformat(mixed daten, mixed abschnitt, mixed
fussnoten, mixed vorschau)
```

leistungsuebersicht() erzeugt die informelle Studenten-Notenuebersicht $daten: kompletter Datensatz eines Studenten (Struktur: s. generiereUebersicht() in notenuebersicht.inc) $abschnitt: auszuwertender Studienabschnitt (VK, K) $fussnoten: anzuzeigende Fussnoten $vorschau: gibt an, ob Wasserzeichen und !Unterschrift auf Seite gedruckt werden soll

---

letzteSucheAnzeigen

bs/blauessystem/suche.php at line 62

```
public void letzteSucheAnzeigen()
```

---

loescheLock

bs/blauessystem/include/general.inc at line 179

```
public void loescheLock(mixed id, mixed art)
```

loescheLock() loescht den Lock auf einen Datensatz $id: die ID des Datensatzes $art: die Art des Locks

---

loeschen

bs/blauessystem/verw13.php at line 176

```
public void loeschen()
```

---

login

bs/blauessystem/index.php at line 39

```
public void login(mixed origin)
```

---

logout

bs/blauessystem/index.php at line 19

```
public void logout()
```

---

noteneintragungRueckgaengig

bs/blauessystem/include/noten.inc at line 14

```
public void noteneintragungRueckgaengig(mixed id, mixed veranstaltungid, mixed
studentid, mixed benutzerid, mixed zeitpunkt, mixed aktion, mixed note)
```

eintragen: aus Veranstaltung austragen (wenn noch eingetragen), Eintrag und alle Ereignisse die danach folgen loeschen aendern: auf vorherige Note zuruecksetzen und evtl. wieder eintragen, Eintrag und alle Ereignisse die danach folgen loeschen austragen: mit vorheriger Note eintragen, Eintrag und alle Ereignisse die danach folgen loeschen

---

notenuebersicht

bs/blauessystem/include/notenuebersicht.inc at line 170

```
public void notenuebersicht(mixed studenten, mixed abschnitt, mixed art, bool
header, bool vorschau, bool print, int start, int limit, int v, bool
gruppierung)
```

Bereitet die Daten fuer die Notenuebersicht vor $studenten: Array mit StudentenIDs (Sortierung ist zu beachten!) $abschnitt: Studienabschnitt (VK, K, VK_K), fuer den die Uebersicht erstellt werden soll $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $header: Gibt an, ob ein pdf-Header erstellt werden soll $vorschau: Gibt an fuer Leistungsnachweis/-uebersicht an, ob es sich um eine Druckvorschau handelt $print: TRUE, wenn die Uebersichten direkt gedruckt werden sollen, ansonsten werden die $daten zurueckgegeben (z.B. fuer Notenspiegel)

---

notenuebersichtQuerformat

bs/blauessystem/include/notenuebersichtQuerformat.inc at line 197

```
public void notenuebersichtQuerformat(mixed studenten, mixed abschnitt, mixed
art, bool header, bool vorschau, bool print, int start, int limit, int v, bool
gruppierung)
```

Bereitet die Daten fuer die Notenuebersicht im Querformat vor $studenten: Array mit StudentenIDs (Sortierung ist zu beachten!) $abschnitt: Studienabschnitt (VK, K, VK_K), fuer den die Uebersicht erstellt werden soll $art: Art der Uebersicht (A4, A5, LPA, LPA2, TOR) $header: Gibt an, ob ein pdf-Header erstellt werden soll $vorschau: Gibt an fuer Leistungsnachweis/-uebersicht an, ob es sich um eine Druckvorschau handelt $print: TRUE, wenn die Uebersichten direkt gedruckt werden sollen, ansonsten werden die $daten zurueckgegeben (z.B. fuer Notenspiegel)

---

notenzeile

bs/blauessystem/include/pdf/tor.inc at line 60

```
public void notenzeile(mixed bezeichnung, mixed kurstyp, mixed stunden, mixed
ects, mixed benotung, str notenzusatz, mixed zeilentyp, mixed spaltentyp, str
vorname, str name, str matrikel)
```

notenzeile() erstellt eine Zeile der Notenuebersicht fuer einen Schein $bezeichnung: Scheintitel $kurstyp: Kurstyp des Scheins $stunden: Gesamtzahl der Stunden des Scheins $ects: ECTS-Punkte zum Schein $benotung: Note des Scheins $notenzusatz: Noten-Ergaenzung a la 'Inland' (*), 'Ausland' (**),... $zeilentyp: Header-Zeile oder normale Zeile? $spaltentyp: Bestimmt die Anzahl der Spalten

---

notenzeileQuerformat

bs/blauessystem/include/pdf/leistungsuebersichtQuerformat.inc at line 32

```
public void notenzeileQuerformat(mixed nr, mixed bezeichnung, mixed
scheinvorlageTitel, mixed benotung, mixed notenzusatz, mixed datum, mixed
semester, mixed zeitraum, mixed scheintyp)
```

notenzeile() erstellt eine Zeile der Notenuebersicht fuer einen Schein $nr: Laufende Nummer des Scheins $bezeichnung: Scheintitel $benotung: Note des Scheins $notenzusatz: Noten-Ergaenzung a la 'Inland' (*), 'Ausland' (**),... $datum: Datum, zu welchem der Schein

ausgestellt wurde $scheintyp: Art des Scheins

---

notizAnlegen

bs/blauessystem/stud01.php at line 238

```
public void notizAnlegen()
```

---

notizBearbeiten

bs/blauessystem/stud01.php at line 187

```
public void notizBearbeiten()
```

---

notizLoeschen

bs/blauessystem/stud01.php at line 215

```
public void notizLoeschen()
```

---

notizSpeichern

bs/blauessystem/stud01.php at line 110

```
public void notizSpeichern()
```

---

pjWahlfachAnlegen

bs/blauessystem/verw12.php at line 19

```
public void pjWahlfachAnlegen(mixed modus)
```

---

pjWahlfachBearbeiten

bs/blauessystem/verw13.php at line 49

```
public void pjWahlfachBearbeiten(mixed id, mixed useDB, mixed modus)
```

---

pjWahlfaecherAnzeigen

bs/blauessystem/verw13.php at line 22

```
public void pjWahlfaecherAnzeigen()
```

---

printArray

bs/blauessystem/include/general.inc at line 133

```
public void printArray(mixed array, bool exit)
```

printArray() gibt den Inhalt eines Array formatiert aus $array: anzuzeigendes Array $exit: wenn TRUE bricht printArray die Ausfuehrung mit exit ab

---

restorePreviousSeitenmodus

bs/blauessystem/include/session.inc at line 150

```
public void restorePreviousSeitenmodus()
```

---

saveSeitenmodus

bs/blauessystem/include/session.inc at line 139

```
public void saveSeitenmodus(mixed seitenmodus)
```

---

scheinAnlegen

bs/blauessystem/verw06.php at line 19

```
public void scheinAnlegen(mixed modus)
```

---

scheinBearbeiten

bs/blauessystem/verw07.php at line 39

```
public void scheinBearbeiten(mixed id, mixed useDB, mixed modus)
```

---

scheindruck

bs/blauessystem/include/scheindruck.inc at line 139

```
public void scheindruck(mixed scheine, mixed semester, bool vorschau, bool
uebersicht, int student, str teilleistung, str veranstaltung)
```

scheindruckTeilleistungen

bs/blauessystem/include/scheindruckTeilleistungen.inc at line 138

```
public void scheindruckTeilleistungen(mixed scheine, mixed semester, bool
vorschau, bool uebersicht, int student, str teilleistung, str veranstaltung)
```

scheineAnzeigen

bs/blauessystem/verw07.php at line 22

```
public void scheineAnzeigen()
```

semesterAnlegen

bs/blauessystem/verw10.php at line 37

```
public void semesterAnlegen(mixed modus)
```

semesterAnzeigen

bs/blauessystem/verw11.php at line 22

```
public void semesterAnzeigen()
```

semesterBearbeiten

bs/blauessystem/verw11.php at line 39

```
public void semesterBearbeiten(mixed id, mixed useDB, mixed modus)
```

semesterDBErstellen

bs/blauessystem/verw10.php at line 20

```
public void semesterDBErstellen(mixed art, mixed jahr, mixed anfang, mixed ende,
mixed vorlesunganfang, mixed vorlesungende)
```

semesterKurz

bs/blauessystem/include/session.inc at line 117

```
public void semesterKurz(mixed jahr, mixed art)
```

semesterLang

bs/blauessystem/include/session.inc at line 107

```
public void semesterLang(mixed jahr, mixed art)
```

session

bs/blauessystem/include/session.inc at line 9

```
public void session()
```

sessionBeenden

bs/blauessystem/include/session.inc at line 65

```
public void sessionBeenden()
```

setPaneInfo

bs/blauessystem/include/tabbedpane.inc at line 16

```
public void setPaneInfo(mixed id)
```

setPaneInfo() holt alle benoetigten Daten des Studenten zur Ansicht in der TabbedPane. $id: Die StudentID

setSuche

bs/blauessystem/include/session.inc at line 135

```
public void setSuche(mixed suche)
```

setzeLock

bs/blauessystem/include/general.inc at line 150

```
public void setzeLock(mixed id, mixed art, mixed benutzer)
```

setzeLock() versucht, einen Lock auf einen Datensatz zu setzen. Der Rueckgabewert gibt dabei an, ob die Sperrung erfolgreich war $id: die ID des Datensatzes $art: die Art des Locks $benutzer: die ID des bearbeitenden Benutzers

---

shownode

bs/blauessystem/verw01.php at line 207

```
public void shownode(mixed x)
```

---

signaturAngehaengt

bs/blauessystem/include/dozent.inc at line 2

```
public void signaturAngehaengt()
```

---

signaturHochladen

bs/blauessystem/verw09.php at line 101

```
public void signaturHochladen()
```

---

signaturLoeschen

bs/blauessystem/verw09.php at line 118

```
public void signaturLoeschen()
```

---

signaturTest

bs/blauessystem/include/dozent.inc at line 7

```
public void signaturTest()
```

---

smartyConstant

bs/blauessystem/include/smarty.inc at line 30

```
public void smartyConstant(mixed key, mixed value)
```

smartyConstant() definiert eine ubergebene Variable als PHP-Konstante und Smarty-Variable $key: Name der Konstante/Smarty-Variable $value: Wert der Konstante/Smarty-Variable

---

smarty_function_button

bs/blauessystem/smarty/plugins/function.button.php at line 12

```
public void smarty_function_button(mixed params, mixed smarty)
```

---

smarty_function_checkbox

bs/blauessystem/smarty/plugins/function.checkbox.php at line 13

```
public void smarty_function_checkbox(mixed params, mixed smarty)
```

---

smarty_function_createLink

bs/blauessystem/smarty/plugins/function.createLink.php at line 14

```
public void smarty_function_createLink(mixed params, mixed smarty)
```

---

smarty_function_dropdown

bs/blauessystem/smarty/plugins/function.dropdown.php at line 12

```
public void smarty_function_dropdown(mixed params, mixed smarty)
```

---

smarty_function_fehler

bs/blauessystem/smarty/plugins/function.fehler.php at line 12

```
public void smarty_function_fehler(mixed params, mixed smarty)
```

---

smarty_function_graphbutton

bs/blauessystem/smarty/plugins/function.graphbutton.php at line 12

```
public void smarty_function_graphbutton(mixed params, mixed smarty)
```

---

smarty_function_hinweis

bs/blauessystem/smarty/plugins/function.hinweis.php at line 12

```
public void
smarty_function_hinweis(mixed params, mixed smarty)
```

---

bs/blauessystem/smarty/plugins/function.kohortenuebersicht.php at line 11

smarty_function_kohortenuebersicht

```
public void smarty_function_kohortenuebersicht(mixed params, mixed smarty)
```

---

bs/blauessystem/smarty/plugins/function.leistungsuebersicht.php at line 13

smarty_function_leistungsuebersicht

```
public void smarty_function_leistungsuebersicht(mixed params, mixed smarty)
```

---

bs/blauessystem/smarty/plugins/function.loadCustomCss.php at line 11

smarty_function_loadCustomCss

```
public void smarty_function_loadCustomCss(mixed params, mixed smarty)
```

---

smarty_function_pouebersicht

bs/blauessystem/smarty/plugins/function.pouebersicht.php at line 13

```
public void smarty_function_pouebersicht(mixed params, mixed smarty)
```

---

smarty_function_radiobutton

bs/blauessystem/smarty/plugins/function.radiobutton.php at line 12

```
public void smarty_function_radiobutton(mixed params, mixed smarty)
```

---

smarty_function_submitbutton

bs/blauessystem/smarty/plugins/function.submitbutton.php at line 12

```
public void smarty_function_submitbutton(mixed params, mixed smarty)
```

---

smarty_function_textarea

bs/blauessystem/smarty/plugins/function.textarea.php at line 12

```
public void smarty_function_textarea(mixed params, mixed smarty)
```

---

smarty_function_textfield

bs/blauessystem/smarty/plugins/function.textfield.php at line 13

```
public void smarty_function_textfield(mixed params, mixed smarty)
```

---

smarty_function_tor

bs/blauessystem/smarty/plugins/function.tor.php at line 12

```
public void smarty_function_tor(mixed params, mixed smarty)
```

---

bs/blauessystem/smarty/plugins/function.uebersichtNoteneintragung.php at line 14

smarty_function_uebersichtNoteneintragung

```
public void smarty_function_uebersichtNoteneintragung(mixed params, mixed
smarty)
```

---

bs/blauessystem/smarty/plugins/function.uebersichtNotiz.php at line 13

smarty_function_uebersichtNotiz

```
public void smarty_function_uebersichtNotiz(mixed params, mixed smarty)
```

---

bs/blauessystem/smarty/plugins/function.uebersichtTeilNoteneintragung.php at line 14

smarty_function_uebersichtTeilNoteneintragung

```
public void smarty_function_uebersichtTeilNoteneintragung(mixed params, mixed
```

```
smarty)
```

---

smarty_function_upload

bs/blauessystem/smarty/plugins/function.upload.php at line 12

```
public void smarty_function_upload(mixed params, mixed smarty)
```

---

smarty_modifier_date_format

bs/blauessystem/smarty/libs/plugins/modifier.date_format.php at line 31

```
public string|void smarty_modifier_date_format(string string, string format,
string default_date)
```

Smarty date_format modifier plugin

Type: modifier
Name: date_format
Purpose: format datestamps via strftime
Input:

- string: input date string
- format: strftime format for output

- default_date: default date if $string is empty

See Also:

date_format (Smarty online manual)

Author:

Monte Ohrt

Uses:

smarty_make_timestamp()

---

bs/blauessystem/smarty/plugins/modifier.escapeJavascript.php at line 13

smarty_modifier_escapeJavascript

```
public void smarty_modifier_escapeJavascript(mixed string)
```

---

smarty_modifier_stripSlashes

bs/blauessystem/smarty/plugins/modifier.stripSlashes.php at line 12

```
public void smarty_modifier_stripSlashes(mixed string)
```

---

smarty_modifier_truncateCenter

bs/blauessystem/smarty/plugins/modifier.truncateCenter.php at line 13

```
public void smarty_modifier_truncateCenter(mixed string, int length, str etc)
```

---

speichereFehler

bs/blauessystem/include/plausichecks.inc at line 265

```
public void speichereFehler(String art, String system, String fehler, integer
scheinID, integer studentID, String message, integer ampelfarbe, boolean
checked)
```

Desc:

Speichert jeden Fehler in der Datenbank ab.

---

speichern

bs/blauessystem/verw13.php at line 110

```
public void speichern()
```

---

standard

bs/blauessystem/verw15.php at line 52

```
public void standard()
```

Desc:

Hier werden alle Scheine geladen, die Studienordnung zugeordnet wurden. Zudem werden die Kriterien geladen.

---

startPlausiChecks

bs/blauessystem/index.php at line 146

```
public void startPlausiChecks()
```

Startet die Plausibilitäts-Checks durch einen AJAX-Request. Dieser ruft die plausicheck.php auf.

---

str_split

bs/blauessystem/include/tcpdf/qrcode.php at line 260

```
public If str_split($string (string), $split_length (int), mixed string, int split_length)
```

Convert a string to an array (needed for PHP4 compatibility)

Parameters:

(string) - The input string.
(int) - Maximum length of the chunk.

Returns:

the optional split_length parameter is specified, the returned array will be broken down into chunks with each being split_length in length, otherwise each chunk will be one character in length. FALSE is returned if split_length is less than 1. If the split_length length exceeds the length of string , the entire string is returned as the first (and only) array element.

---

studentAnlegen

bs/blauessystem/verw04.php at line 19

```
public void studentAnlegen(mixed modus)
```

---

studentBearbeiten

bs/blauessystem/stud06.php at line 22

```
public void studentBearbeiten(mixed id, mixed useDB, mixed modus)
```

---

studentenAnzeigen

bs/blauessystem/suche.php at line 24

```
public void studentenAnzeigen(mixed suche)
```

studentenAnzeigen($suche) zeigt die Ergebnisse der Studentensuche an $suche: Der Suchstring

---

studentendaten

bs/blauessystem/include/pdf/tor.inc at line 18

```
public void studentendaten(mixed key, mixed value)
```

studentendaten() erzeugt die zeilenweisen Studentendaten $key: Beschreibung des Datums $value: Datums-Wert

---

studentendetails

bs/blauessystem/stud02.php at line 79

```
public void studentendetails(mixed studentID, mixed aktSemester)
```

studentendetails() zeigt die Detailansicht (nach Semestern sortierte Scheine + FLNs) eines Studenten an $studentID: anzuzeigender Student $aktSemester: anzuzeigendes Semester

---

toggleNavVisible

bs/blauessystem/include/session.inc at line 200

```
public void toggleNavVisible(mixed id)
```

---

tor

bs/blauessystem/include/pdf/tor.inc at line 207

```
public void tor(mixed daten, mixed fussnoten, mixed noten, mixed kurstypen, bool vorschau)
```

tor() erzeugt den transcript of records $daten: kompletter Datensatz eines Studenten (Struktur: s. generiereUebersicht() in notenuebersicht.inc) $fussnoten: anzuzeigende Fussnoten $noten: Zuordnungen dt. Noten - intl. Noten $kurstypen: moegliche Kurztypen und deren dt. und intl. Bezeichnung $vorschau: gibt an, ob eine Vorschau des ToR gedruckt werden soll (noch nicht verwendet)

---

torAnzeigen

bs/blauessystem/stud04.php at line 17

```
public void torAnzeigen(mixed studentid)
```

---

torFooter

bs/blauessystem/include/pdf/tor.inc at line 172

```
public void torFooter()
```

torFooter() erzeugt die Fusszeile des Transcript of Records (Seitenanzahl)

---

torFussnoten

bs/blauessystem/include/pdf/tor.inc at line 186

```
public void torFussnoten(mixed fussnoten, mixed abstand)
```

torFussnoten() erzeugt die zu den Notenuebersichten gehoerenden Fussnoten $fussnoten: Anzuzeigende Fussnoten

---

torHeader

bs/blauessystem/include/pdf/tor.inc at line 154

```
public void torHeader(mixed vorname, mixed name, mixed matrikel)
```

torHeader() erzeugt die Kopfzeile des Transcript of Records (Uni-Logo und Student) $vorname: Vorname des Studenten $name: Name des Studenten $matrikel: Matrikelnummer des Studenten

---

uebersicht

bs/blauessystem/note05.php at line 52

```
public void uebersicht()
```

Desc:

Eine Prüfungsordnung wurde ausgewählt. Nun die Übersichtstabelle für Vorklinik und Klinik inkl. der Kritieren laden.

---

uebersichtAnzeigen

bs/blauessystem/stud03.php at line 17

```
public void uebersichtAnzeigen(mixed studentid, mixed abschnitt)
```

---

uebersichtK

bs/blauessystem/stud03.php at line 54

```
public void uebersichtK()
```

---

uebersichtVK

bs/blauessystem/stud03.php at line 45

```
public void uebersichtVK()
```

---

umleiten

bs/blauessystem/include/session.inc at line 70

```
public void umleiten(str origin)
```

---

update

bs/blauessystem/verw15.php at line 171

```
public void update()
```

Desc:

Die Importkriterien eines Scheines wurden bearbeitet und werden nun gespeichert. Dazu zunächst die alten Kriterien löschen und die neuen einfügen.

---

updatePO

bs/blauessystem/verw14.php at line 195

```
public void updatePO()
```

---

veranstaltungAnlegen

bs/blauessystem/verw01.php at line 20

```
public void veranstaltungAnlegen(mixed modus)
```

---

veranstaltungBearbeiten

bs/blauessystem/verw02.php at line 54

```
public void veranstaltungBearbeiten(mixed id, mixed useDB, mixed modus)
```

---

veranstaltungenAnlegen

bs/blauessystem/verw10.php at line 95

```
public void veranstaltungenAnlegen()
```

---

veranstaltungenAnzeigen

bs/blauessystem/verw02.php at line 22

```
public void veranstaltungenAnzeigen()
```

---

zeigeCSVUpload

bs/blauessystem/note04.php at line 287

```
public void zeigeCSVUpload(String aktVeranstaltung:, mixed aktVeranstaltung)
```

Desc:

Holt zu einer ausgewählten Veranstaltung alle zugeordneten Teilleistungen und zeigt diese in einer Auswahlliste an.

Parameters:

aktVeranstaltung: - Veranstaltung, bei der die Noten eingetragen werden sollen

---

zeigeImport

bs/blauessystem/verw03.php at line 37

```
public void zeigeImport()
```

zeigeImport() holt sucht die Studienordnungen heraus und zeigt die Startseite des Studentenimports an

---

zeigeSeite

bs/blauessystem/stud02.php at line 28

```
public void zeigeSeite(str fehler)
```

---

zeitspanneInMonaten

bs/blauessystem/include/plausichecks.inc at line 289

```
public void zeitspanneInMonaten(String datumErstePruefung, String
datumZweitePruefung)
```

Desc:

Diese Methode gibt die Differenz zwischen 2 Daten zurück

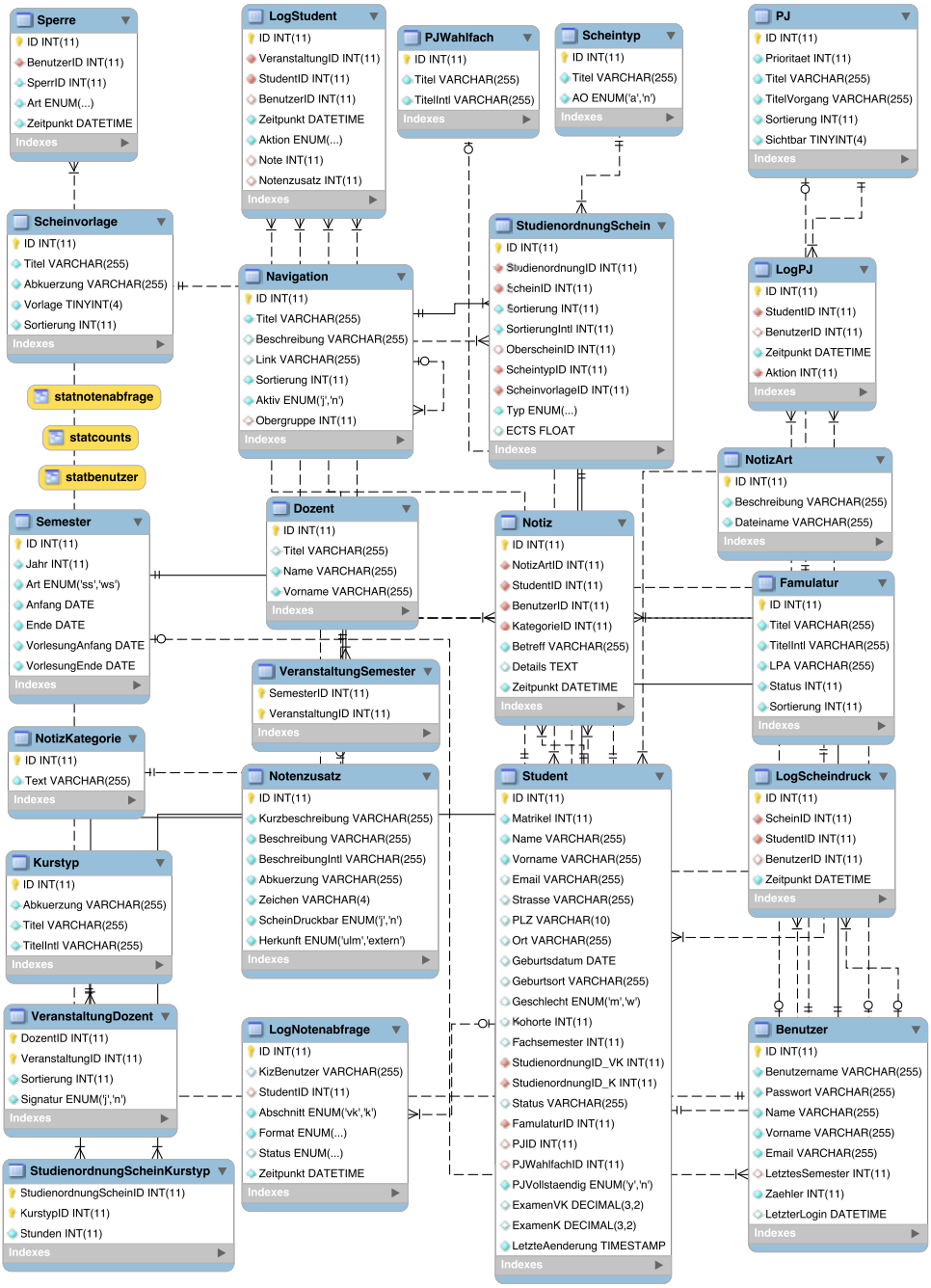


Abbildung 9.1: Die Gesamtübersicht der Datenbank

Literaturverzeichnis

[1] Studiendekanat Medizin. Blaues System. (Last visited: 05.12.2012) Website. [Online]. Available: https://blau.medizin.uni-ulm.de/~leif/blau/

[2] Hochschulinformationssystem. (Last visited: 05.11.2012) Website. [Online]. Available: http://de.wikipedia.org/wiki/Hochschulinformationssystem

[3] Hochschul-Informations-System. (Last visited: 06.11.2012) Website. [Online]. Available: http://de.wikipedia.org/wiki/Hochschul-Informations-System

[4] Helpdesk Universität Ulm. Dienstleistungen für die Universitätsverwaltung. (Last visited: 06.11.2012) Website. [Online]. Available: http://www.uni-ulm.de/einrichtungen/kiz/it/dienste-fuer-die-verwaltung.html

[5] Tom Karasek, "HIS-Software-Produkttelegramme," PDF, März 2012.

[6] Amt für amtliche Veröffentlichungen der Europäischen Gemeinschaften, *ECTS-Leitfaden*, Europäische Kommision, 2009.

[7] HIS Hochschul-Informations-System GmbH. Über HIS. (Last visited: 06.11.2012) Website. [Online]. Available: http://www.his.de/unternehmen

[8] HIS Hochschul-Informations-System GmbH. Nutzer. (Last visited: 06.11.2012) Website. [Online]. Available: http://www.his.de/partner

[9] "CampusNet - Das integrierte Campus-Management-System," PDF, Datenlotsen Informationssysteme GmbH, März 2012.

[10] "Kein Interesse an der Katze im Sack?" PDF, Datenlotsen Informationssysteme GmbH, November 2011.

[11] Michael Leander Schrei, Boris Rohrbacher, Ed., *Identity Management – Best Practice*, vol. 73, no. 19. Bonn: Jan von Knop, Wilhelm Haverkamp, Eike Jessen, 2005.

[12] Technische Universität Graz. Funktionskatalog. (Last visited: 08.11.2012) Website. [Online]. Available: http://campusonline.tugraz.at/campusonline/funktionskatalog

[13] Technische Universität Graz. Kooperationspartner. (Last visited: 06.11.2012) Website. [Online]. Available: http://campusonline.tugraz.at/category/co-operation

[14] Marcel Minke, *Entwurf und Implementierung eines Online-Klausuranmeldesystems unter besonderer Berücksichtigung datenschutzrechtlicher Aspekte*. GRIN Verlag, 2005.

[15] Thomas Lux, *Intranet Engineering: Einsatzpotenziale und phasenorientierte Gestaltung eines sicheren Intranet in der Unternehmung*. Springer, 2005.

[16] Universität Bamberg. FlexNow - Architektur. (Last visited: 18.11.2012) Website. [Online]. Available: http://flexnow.uni-bamberg.de/index.php?content=architektur

[17] Universität Bamberg. FlexNow - Funktionalität. (Last visited: 18.11.2012) Website. [Online]. Available: http://flexnow.uni-bamberg.de/index.php?content=funktionalitaet

[18] Elmar J. Sinz, "Prozeßgestaltung und Prozeßunterstützung im Prüfungswesen," in *Informationssysteme für das Hochschulmanagement*, 1997, pp. 121–132.

[19] Universität Bamberg. FlexNow - Aktueller Stand. (Last visited: 18.11.2012) Website. [Online]. Available: http://flexnow.uni-bamberg.de/index.php?content=aktuell

[20] PRIMUSS Campus IT. (Last visited: 09.11.2012) Website. [Online]. Available: http://de.wikipedia.org/wiki/PRIMUSS_Campus_IT

[21] Hochschule für angewandte Wissenschaften FH Ingolstadt. Der Primuss Verbund. (Last visited: 09.11.2012) Website. [Online]. Available: http://www.primuss.de/primuss.html

[22] Hochschule für angewandte Wissenschaften FH Ingolstadt. PRIMUSS - Datenfluss. (Last visited: 09.11.2012) Website. [Online]. Available: http://www.primuss.de/technik.html

[23] Hochschule für angewandte Wissenschaften FH Ingolstadt. PRIMUSS - Aufteilung in Module. (Last visited: 09.11.2012) Website. [Online]. Available: http://www.primuss.de/module.html

[24] Hochschule für angewandte Wissenschaften FH Ingolstadt. PRIMUSS - IT-Campus-Management. (Last visited: 09.11.2012) Website. [Online]. Available: http://www.primuss.de/campus-mangement.html

[25] FH-Complete. (Last visited: 15.11.2012) Website. [Online]. Available: http://de.pluspedia.org/wiki/FH-Complete

[26] GNU General Public License. (Last visited: 15.11.2012) Website. [Online]. Available: http://de.wikipedia.org/wiki/Gpl

[27] FH Technikum Wien, Abteilung Infrastruktur, "FH COMPLETE - Das Software-Paket für Hochschulen," PDF, FH Technikum Wien, Höchstädtplatz 5, 1200 Wien, 2011.

[28] eLeDia - E-Learning im Dialog GmbH. Moodle. (Last visited: 18.11.2012) Website. [Online]. Available: http://moodle.de/

[29] ILIAS open source e-Learning e.V. Ilias. (Last visited: 18.11.2012) Website. [Online]. Available: http://www.ilias.de

[30] Bogdan Cristea, Shaffin Bhanji. FreeSMS (Free Student Management System). (Last visited: 05.12.2012) Website. [Online]. Available: http://sourceforge.net/projects/freesms/

[31] FH Technikum Wien. FH Complete. (Last visited: 01.12.2012) Website. [Online]. Available: http://fhcomplete.technikum-wien.at/index.html

[32] Dipl.-Päd. Claudia Grab, "Studienführer Humanmedizin," PDF, Medizinische Fakultät der Universität Ulm, Albert-Einstein-Allee 7, 89081 Ulm, September 2011.

[33] Universität Ulm. Studiengang Molekulare Medizin. (Last visited: 29.11.2012) Website. [Online]. Available: http://www.uni-ulm.de/studium/studiengaenge/bachelorstudiengaenge/medizin/molekulare-medizin-bachelor.html

[34] Barbara Eichner, Stephanie Hinderberger, "Studiengang Molekulare Medizin (Bachelor, Master)," PDF, Universität Ulm, 89069 Ulm, Juni 2012.

[35] Senat Uni Ulm, "Studienordnung der Universität Ulm bis zum Ersten Abschnitt der Ärtzlichen Prüfung des Studiengangs Humanmedizin (Vorklinik)," PDF, August 2005.

[36] GNU Lesser General Public License. (Last visited: 12.11.2012) Website. [Online]. Available: http://de.wikipedia.org/wiki/GNU_Lesser_General_Public_License

[37] Monte Ohrt and Messju Mohr. Smarty. (Last visited: 12.11.2012) Website. [Online]. Available: http://de.wikipedia.org/wiki/Smarty

[38] Nauman, Ryan, "Getting started with PHP and smarty," *Crossroads*, vol. 14, no. 3, pp. 9–16, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1373576.1373579

[39] Studiendekanat Medizin. Installation & Wartung. (Last visited: 13.11.2012) Website. [Online]. Available: https://studiendekanat.medizin.uni-ulm.de/doku/doku.php?id=softwaresysteme:blauessystem:installation_wartung

[40] Tcpdf. (Last visited: 13.11.2012) Website. [Online]. Available: http://en.wikipedia.org/wiki/TCPDF

[41] Nicola Asuni. Tcpdf. (Last visited: 13.11.2012) Website. [Online]. Available: http://www.tcpdf.org/

[42] *Evaluation von Lehrveranstaltungen effizient umsetzen*, Electric Paper Evaluationssysteme GmbH, Konrad-Zuse-Allee 13, 21337 Lüneburg, September 2012.

[43] Manfred Reichert and Rüdiger Pryss. MedicUlm. (Last visited: 20.11.2012) Website. [Online]. Available: http://www.uni-ulm.de/in/iui-dbis/news-einzelansicht/article/dbis-entwickelt-iphone-und-android-app-fuer-die-medizinische-fakultaet-der-universitaet-ulm.html

[44] kiz. E-Learning mit ILIAS und Moodle. (Last visited: 06.12.2012) Website. [Online]. Available: http://www.uni-ulm.de/einrichtungen/kiz/it/campus-systeme/elearning.html

[45] Fabian Schwarz, “Untersuchung der Augmented Reality Technik am Beispiel des Android Frameworks,” Bachelor Thesis, September 2011.

[46] Michaela Jaeger, “Mobile Web Service Entwicklung mittels Android 2.1,” Bachelor Thesis, February 2011.

[47] ksoap2-android. (Last visited: 23.11.2012) Website. [Online]. Available: http://code.google.com/p/ksoap2-android/

[48] Andreas Kilian, “Metadaten-basierter Aufgabenplaner meTask für den mobilen Einsatz am Beispiel des Android Frameworks,” Bachelor Thesis, September 2011.

[49] Martin Fröchtenicht, “Social Application Wazzup,” Bachelor Thesis, September 2011.

[50] Kent Ka Iok Tong, *Developing Web Services with Apache CXF and Axis2*. TipTec Development, Januar 2010, no. 3.

[51] Srinath Perera, Chathura Herath, Jaliya Ekanayake, “Axis2, Middleware for Next Generation Web Services,” in *International Conference on Web Services, 2006. ICWS ’06*, 2006, pp. 833 –840.

[52] Manfred Reichert and Barbara Weber, *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Berlin-Heidelberg: Springer, 2012.

Abbildungsverzeichnis

Name: Rainer Mösle Matrikelnummer: 593390

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Rainer Mösle