Hochschule Neubrandenburg
University of Applied Sciences

Fachbereich: Landschaftsarchitektur, Geoinformatik,
Geodäsie, Bauingenieuerwesen

Studiengang Geoinformatik

Bachelorarbeit
zum Thema

Entwicklung einer Anbindung von Mobilfunkgeräten an einen Online-Stundenplan

Zum Erlangen des akademischen Grades
„Bachelor of Engineering“ (B.Eng.)

vorgelegt von: Jörg Blaufuß
urn:nbn:de:gbv:519-thesis2009-0022-8

Erstprüfer: Herr Prof. Dr.-Ing. Andreas Wehrenpfennig
Zweitprüfer: Herr Dr. rer. nat. Martin Nitschke

Bearbeitungszeitraum: 27. März 2009 - 22.Mai.2009

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neubrandenburg, den 22. Mai 2009

Jörg Blaufuß

Kurzfassung

Im Rahmen dieser Bachelorarbeit wurde ein Konzept entwickelt, um die Daten des Online-Stundenplans der Hochschule so aufzuarbeiten, dass die Studierenden die Informationen per Mobilfunkgerät abfragen können. Dabei werden verschiedene Verfahren erläutert mit denen der Entwurf realisiert werden kann. Die prototypische Umsetzung basiert auf dem Wireless Application Protocol (WAP) und ist so angelegt, dass sie auf den für die Hochschule Neubrandenburg entwickelten Stundenplan abgestimmt ist und darüber hinaus die Möglichkeit bietet durch einfache Änderungen an andere Systeme angepasst zu werden.

Abstract

Within the scope of this bachelor thesis a concept was developed to handle the data of the online-timetable for getting the information by mobile phone. Thereby different processes are exemplified. The prototypical design based on the technology of the wireless application protocol (WAP), is build up for the developed timetable of the University of Applied Science Neubrandenburg and gives the possibility to adapt on other systems by minimal modification.

Inhaltsverzeichnis

1. Einleitung in die Thematik

Das Mobiltelefon ist heutzutage nicht mehr aus unserer Gesellschaft wegzudenken. Ursprünglich entwickelt um an jedem Ort erreichbar zu sein, verfügt das Handy mittlerweile über eine Vielzahl von nützlichen Funktionen und Erweiterungen, die über das Telefonieren hinausgehen. Integrierte Kameras, Radios, Organizer, Datenspeicher und andere technische Innovationen bieten viele Nutzungsmöglichkeiten in einem Gerät.
Auch steigt die Beliebtheit bei der Nutzung des mobilen Internets womit es den Nutzern möglich ist, auch Unterwegs auf Online-Inhalte zuzugreifen.

Das Ziel dieser Bachelorarbeit ist es, ein Konzept zur Aufbereitung der Daten eines Online-Stundenplans zu entwickeln, welches den Studierenden ermöglichen soll die Informationen per Mobiltelefon abzufragen. Dabei ist darauf zu achten, die Lösung allgemein zu halten um mittels minimaler Schnittstelle an verschiedene Systeme andocken zu können.
Der prototypische Entwurf orientiert sich an dem von der Hochschule Neubrandenburg entwickelten Stundenplansystem und soll die Funktionalität der Schnittstelle zwischen dem World Wide Web (WWW) und dem mobilen Endgerät aufzeigen.

Mit der Arbeit wird die Grundlage für eine Funktion geschaffen, mit der die Studierenden der Hochschule Neubrandenburg zukünftig auch mobil den Stundenplan erreichen können.

2. Grundlagen

2.1. genutzte Stundenplansysteme

Derzeit ist an der Hochschule Neubrandenburg kein einheitliches System für einen Onlinestundenplan im Einsatz. Die Fachbereiche regeln die Auskunft der Vorlesungs-, Seminar- und Übungstermine in Eigenregie und stellen weitestgehend keine, beziehungsweise sehr allgemein gehaltene Onlineinformationen zur Verfügung. Lediglich der Fachbereich Landschaftsarchitektur, Geoinformatik, Geodäsie, Bauingenieurwesen (LGGB) bietet den Studierenden ein umfangreiches webbasiertes System an. Mit Hilfe von verschiedenen Filtern ist es den Lehrenden und Studenten möglich, gezielte Informationen über Termine, Lehrveranstaltungen, Dozenten und Raumbelegungen abzufragen. Wichtige Termine und Änderungen werden farbig hervorgehoben um dem Betrachter darauf aufmerksam zu machen. Im Zuge der Forderung, dass alle Abwicklungen der Onlinedienste über das Onlineportal der Hochschule stattfinden sollen, wurde im Rahmen von Projektarbeiten ein Stundenplansystem entwickelt, welches auf dem bestehenden System des Fachbereichs LGGB basiert und den Anforderungen aller Fachbereiche entspricht. Diese Entwicklung bringt zusätzliche Optionen für die Suche mit, bietet Schnittstellen für weitere Funktionalitäten und gestaltet sich, durch das integrierte Rechtesystem und den damit verbundenen Zugriffsmöglichkeiten für die Dozenten, weitaus dynamischer als der Vorgänger.
Während der Testphase wurde allerdings beschlossen ein anderes System für die Verwaltung einzuführen. Die Hochschul-Informations-System-GmbH (HIS) hat ein Modul für Web-Anwendungen im Bereich Lehre, Studium und Forschung (LSF) entwickelt, welches zukünftig auch Einsatz an der Hochschule Neubrandenburg finden soll. Das System bietet weitreichende Funktionen um Projekte und Veranstaltungen, insbesondere Lehrveranstaltungen, zu erfassen und ermöglicht den Studierenden ihren individuellen Stundenplan zusammenzustellen. Damit steht ein, für alle Fachbereiche einheitliches, System zur Verfügung welches derzeit Bundesweit an 23 weiteren Hochschulen Einsatz findet [1].

2.2. Anwender

In Deutschland waren im Jahr 2008 gesamt 107,4 Millionen Mobilfunkanschlüsse registriert, womit statistisch gesehen jeder fünfte Deutsche 2 Verträge besitzt. 16 Millionen Anschlüsse verfügen dabei zusätzlich über eine Anbindung an UMTS (Universal Mobile Telecommunications System). Es wird erwartet, dass die Zahl der schnellen Internetverbindungen im Jahr 2009 auf 22,7 Millionen ansteigt [2].
Daraus lässt sich erkennen, dass das Interesse an mobilen Datendiensten starken Zuwachs hat. Als Gründe hierfür sind die Entwicklung der Technik, das ständig steigende Angebot an Dienstleistungen sowie eine attraktive Gestaltung der Verbindungspreise, wie zum Beispiel Pauschalpreise für die Internetnutzung („Flatrates“).

Damit ein Datendienst von der Zielgruppe angenommen wird, sind Überlegungen zum möglichen Nutzerverhalten anzustellen. Nützlich sind hierbei Meinungsumfragen auf der Webseite, Gespräche mit Kunden oder Erfahrungsaustausch mit anderen Anbietern. Dabei geben substantielle Fragen über den Nutzer, beziehungsweise dessen Verhalten Aufschluss über Art, Umfang und Notwendigkeit der mobilen Dienste.
Beispiele für solche Fragen sind: „Was wird erwartet?“, „Was ist der Nutzer bereit zu leisten?“ und „Wird der Dienst überhaupt benötigt?“.
Die daraus gezogenen Schlussfolgerungen sind dann Hilfreich um das Angebot den Bedürfnissen der Anwender anzupassen.

2.3. Hardware

Neben dem Mobiltelefon gibt es eine Reihe weiterer Kleingeräte. Hierzu gehören unter anderem kleine Fernsehgeräte, Taschencomputer (PDAs), Pager, Navigationssysteme und Spielkonsolen. Diese sind in der Lage Informationen über die integrierten Schnittstellen (WAP, SMS, Funk) zu empfangen und diese auch darzustellen. Die im weiteren Verlauf genutzte Bezeichnung des Mobiltelefons steht dabei stellvertretend für diese Geräte, da es sich dabei um das am weitesten verbreitete Mittel zur mobilen Datenabfrage handelt.

2.4. Möglichkeiten der mobilen Anbindung

Durch die Vielseitigkeit und den technischen Fortschritt der gängigen Mobilfunkgeräte ergeben sich mehrere Ansätze zu Lösung der Problemstellung. Die folgenden Ausführungen geben eine Übersicht über verschiedene Möglichkeiten zur mobilen Abfrage von Webinhalten.

2.4.1. Sprachdialogsystem

Das Sprachdialogsystem bietet dem Anwender die Möglichkeit, Inhalte von Webseiten über das Telefon abzufragen. Hierzu benötigt der Nutzer lediglich einen Telefonanschluss und ist weder auf einen Computer, noch auf ein Mobiltelefon mit Internetfunktion angewiesen.

Der Nutzer ruft den vom Anbieter bereitgestellten Telefonanschluss an. Mittels Sprachdialog erfolgt die Navigation zu den erwünschten Informationen. Dies kann über verschiedene Verfahren erfolgen. Zu einem besteht die Möglichkeit des vollautomatisierten Dialoges wobei der Anrufer viele Informationen in ganzen Sätzen einbauen kann. Für die Abfrage des Stundenplanes wäre zum Beispiel „Stundenplan für Geoinformatik im 3. Semester am 12.03.2009 um 10 Uhr für Gruppe C“ denkbar. Allerdings müssen hierbei sehr viele Annahmen getroffen werden, da jeder Anrufer ein anderes Sprachverhalten besitzt und dadurch auf jede Situation, wie fehlende oder unnötige Informationen sowie undeutlicher Ausdruck, reagiert werden muss.
Als weitere Möglichkeit bietet sich ein teilautomatisierter Dialog an. Die Navigation erfolgt dabei über die Einzelworterkennung, welche den Vorteil bringt vom Anwender gezielte Informationen abzufragen. Dieses Gespräch könnte im Bezug auf die Stundenplanabfrage wie Folgt aussehen:

Frage: „Welchem Studiengang gehören Sie an?“	Antwort: „Geoinformatik“
Frage: „Welches Semester?“	Antwort: „Drittes“
Frage: „Welche Gruppe?“	Antwort: „C“

Auch hierbei ist wieder eine deutliche Aussprache Voraussetzung. Je nach Fülle der Informationen kann dieser Dialog zeitaufwendig und damit für den Anrufer kostenintensiv werden.

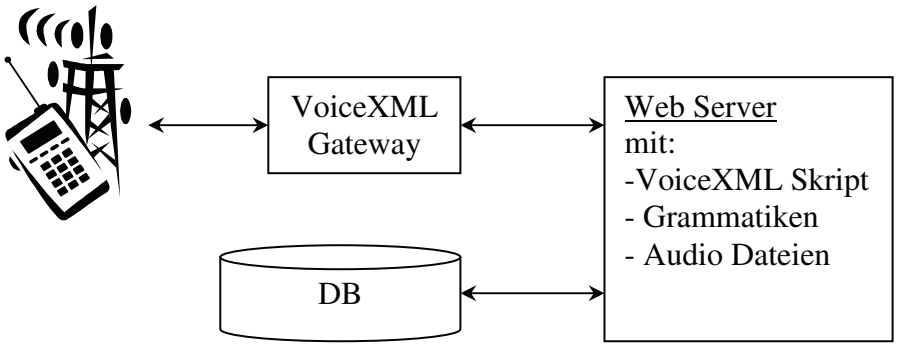


Abbildung 1: Ausführungsbeispiel für VoiceXML-Abfrage

Ganz auf Spracheingabe verzichtet das „Dual-Tone multifrequency dialing (DTMD)“. Die Dialogführung erfolgt seitens des Anbieters über Sprachausgabe und seitens des Anwenders ausschließlich über die Wähltasten des Telefons.
„Für Soziale Arbeit drücken Sie bitte die 1, für Lebensmitteltechnologie die 2 für Geoinformatik die 3 …!“, „Geben sie bitte die Zahl Ihres Semesters ein!“
Der Vorteil hierbei ist, dass Verständnisprobleme ausgeschlossen werden können. Allerdings können die Dialoge sehr lang werden und jede Stufe ist auf maximal 10 Optionen (Tasten 0 – 9) beschränkt.

Das Sprachdialogsystem bietet dem größten Anteil der Hochschulangehörigen die Möglichkeit, den Onlinestundenplan auch mobil zu erreichen wozu lediglich ein Telefonanschluss benötigt wird. Allerdings entstehen für jeden Anruf Kosten, welche anhand der unterschiedlichen Dialoglängen variieren. Auch besteht keine direkte Möglichkeit die Abfragen zu speichern. Die Ausgabe erfolgt im gesprochenen Text was im Vergleich zu einer graphischen Ausgabe den Nachteil hat das die Informationen flüchtig sind und einzig im Kurzzeitgedächtnis verbleiben.
Dieses Verfahren zur Informationsabfrage ist mittlerweile weit verbreitet und findet mannigfache Anwendung. Viele Unternehmen, mit Telefonservice nutzen die interaktive Dialogführung um die große Anzahl von Kundenanfragen bewältigen zu können und auch um auf diesem Wege Kosten für Personal einzusparen. Zu finden ist der Service zum Beispiel für Reiseauskünfte bei diversen Verkehrsunternehmen, als automatische Vermittlung bei Behörden und Ämter sowie in Callcenter jeglicher Art.

2.4.2. Abfrage von Web-Inhalten über SMS-Versand

Der Kerngedanke dieses Verfahrens liegt für den Anwender darin, Zugriff auf bestimmte Inhalte von Webseiten im WAP, Internet und Intranet mittels Short Message Service (SMS) zu erhalten. Hierzu hat der Mobilfunkbetreiber T-Mobile, eine Tochtergesellschaft der Deutschen Telekom, ein Patent [1] angemeldet, welches das Ziel verfolgt den Benutzern handelsüblicher Mobilfunkgeräte, auch ohne WAP-Funktion, diese Informationsabfrage zu gewährleisten [3].

Der Mobilfunkteilnehmer sendet eine SMS mit der jeweiligen Suchanfrage an einen Dienstanbieter, dem so genannten SMS-Dienstezentrum. Der Inhalt der Nachricht wird dort erkannt und an die Schnittstelle zwischen Mobilfunknetz und Internet, dem SMS-Gateway, weitergeleitet. Im Gateway erfolgen dann die automatische Auswertung des Inhalts, die Umformulierung in eine dem Internet verständliche Anfrage und die Weiterleitung der Anfrage an den jeweiligen Informationsprovider. Die dort bereitgestellten Informationen, welche zum Beispiel aus Textausschnitten von HTML- oder WAP-Seiten bestehen können, werden zurück an das Gateway gesendet, dort in das SMS-Format umgewandelt und als eine oder mehrere SMS über das SMS-Dienstzentrum an das Mobilfunkgerät des Teilnehmers übermittelt. In der Abbildung 2 ist ein Ausführungsbeispiel dargestellt, um diesen Ablauf näher zu erläutern.

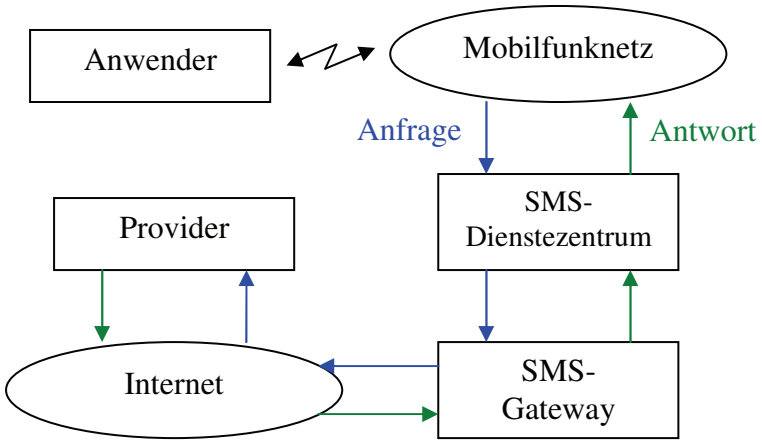


Abbildung 2: Ausführungsbeispiel für SMS-Anfrage

[1] Eingereicht am 12.November 2001 beim Deutsches Patent- und Markenamt, Aktenzeichen: 101 55 197.5

Der Inhalt und Aufbau der jeweiligen Anfrage richtet sich nach den vom Provider bereitgestellten Datensätzen. Eine Möglichkeit hierbei ist eine Volltextsuche, basierend auf der Methode derzeitig verwendeter Internetsuchmaschinen, wobei das Resultat mit den besten Treffern die Antwort ist. Auch können die Nachrichten mit bereits vordefinierter Syntax gesendet werden. Dabei wird die Anfrage ohne Umwandlung durch das Gateway direkt an den Provider übermittelt und verstanden. Dies könnte im Bezug auf eine mobile Abfrage des Stundenplans wie folgt aussehen:
GI 3 C 220609 1000 *für Geoinformatik, 3. Semester, Gruppe C am 22.06.2009 um 10 Uhr*
Denkbar ist auch ein kontextspezifischer Dialog zwischen dem Nutzer und dem Gateway. Das Gateway speichert die Anfrage und sendet gegebenenfalls eine Rückfrage an den Anwender, um die Suchoptionen zu Konkretisieren. Die Anfrage des Anwenders würde sich in diesem Fall auf den Stundenplan für den jeweiligen Studiengang beziehen. Die Antwort des Gateways verlangt dann nach weiteren Informationen, wie z.B. das Semester, die Gruppe und das gewünschte Datum.

Da standardmäßig jedes Mobiltelefon mit SMS-Funktion ausgerüstet ist, deckt dieses Verfahren eine große Benutzergruppe ab. Allerdings hängt das Antwortergebnis von der Qualität der Anfrage ab. Im Fall der Volltextsuche müssen bereits bei der Entwicklung eine Vielzahl von Annahmen getroffen werden, damit die Auswertung korrekt erfolgen kann. Die vordefinierte Syntax wiederum bietet eine gute Lösung für die einfache Suche, kann jedoch für detailreiche Suchanfragen sehr umfangreich und kryptisch ausfallen und wird somit schnell fehleranfällig. Außerdem ist die Anfrage auf die Länge einer SMS, also 160 Zeichen, begrenzt.
Ein großer Nachteil bei dieser Methode sind die anfallenden Kosten. Da es sich hierbei um Servicerufnummern handelt, ist die Gebühr für eine Anfrage höher als für eine Standard-SMS und kann, je nach SMS-Dienstanbieter, variieren. Dem Anbieter des Services entstehen hierbei Kosten für die Nutzung des Gateways und der Datenpflege.

Eingesetzt werden derartige SMS-Dienste derzeit zum Beispiel bei der Onlinesuchmaschine von Google.Inc als lokale Branchensuche, mobiles Lexikon und Glossar. Auch greifen Anbieter von so genannten Fun-Applikationen für das Handy darauf zurück.

2.4.3. Wireless Application Protocol (WAP)

Hinter dem Wireless Application Protocol (kurz WAP) verbergen sich Dienste, welche es ermöglichen sollen Webinhalte auf dem Mobiltelefon lesbar darstellen zu können. Herkömmliche Webseiten können auf kleinen Endgeräten mit begrenztem Display schlecht, beziehungsweise gar nicht dargestellt werden. Die graphische Auflösung der Inhalte ist oft zu groß, die relativ niedrige Speicherkapazität der Geräte sowie die geringe Übertragungsrate sind unzureichend für die umfangreichen Inhalte der Seiten des World Wide Web (WWW).
Auch treten Probleme bei der Verwendung der Auszeichnungssprachen auf. Diese enthalten viele Daten, welche für die Lesbarkeit notwendig sind und benötigen zusätzlichen Speicherplatz. Um dem entgegenzuwirken wurde die Wireless Markup Language (WML), als stark vereinfachte Form des Web-Standards HTML (Hypertext Markup Language) entwickelt und größtenteils bis zum WAP 1.2 Standard eingesetzt. Im Zuge der Einführung des World Wide Web Consortium (W3C)-Standards XHTML (Extensible Hypertext Markup Language) wurde auch eine weitere Lösung für minimalistische Geräte entwickelt. XHTML Basic und XHTML Mobile Profile werden, unter Verwendung grundlegender Module von XHTML, als Standard ab WAP 2.0 eingesetzt um WML langfristig zu ersetzen [4]. Auch sind einige mobile Geräte mit WAP 1.2 bereits in der Lage XHTML-Seiten anzuzeigen.

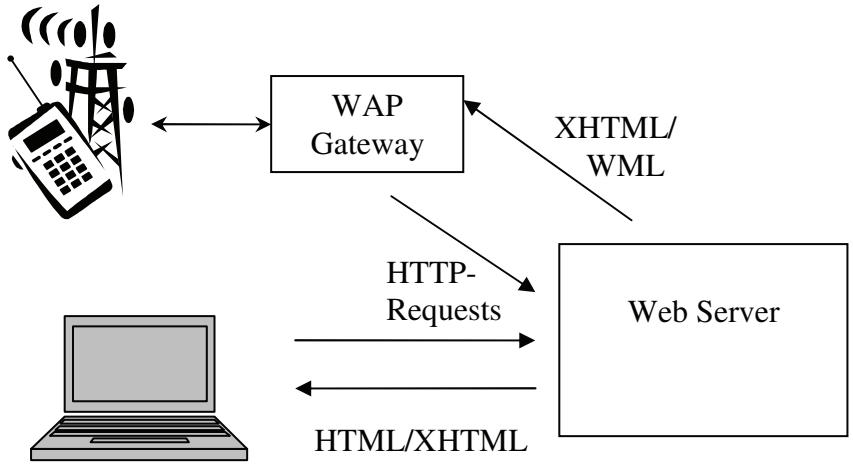


Abbildung 3: Ausführungsbeispiel für WAP

Der Ablauf einer WAP-Anfrage ähnelt dem einer normalen Internetanfrage. Ein zwischengeschaltetes Gateway sorgt dafür, dass die WAP-Requests nach den Richtlinien des HTTP (Hypertext Transfer Protokoll) umgewandelt werden um für den Webserver verständlich zu sein und umgekehrt, den HTTP-Response in ein für den WAP-Client verständliches Format zu konvertieren. In der Abbildung 3 ist ein Ausführungsbeispiel für eine WAP dargestellt.

Ein großer Vorteil bei diesem Verfahren ist, dass als Grundlage die Datenquelle der bereits bestehenden Internetpräsenz genutzt werden kann. Es muss lediglich ein neues Skript in einer für Mobilgeräte geeigneten Sprache angelegt werden. Dabei sollte darauf geachtet werden, dass die darzustellenden Informationen so einfach wie möglich gehalten werden um lange Ladezeiten und Unübersichtlichkeit zu vermeiden. Auch muss bedacht sein, dass die Navigation den Eingabebedingungen der mobilen Geräte angepasst wird. Ist dabei viel Texteingabe erforderlich, wird die Bedienung für den Handynutzer kompliziert und unvorteilhaft, da die Eingabe über die Wähltasten des Telefons erfolgt.
Ein weiterer Nachteil hierbei sind die anfallenden Kosten für die Internetverbindung. Diese variieren ja nach Anbieter und werden, entweder nach Onlinezeit oder durch den Umfang der übertragenen Daten ermittelt.
Vorrangig Webseiteninhaber mit großem Besucheraufkommen bieten neben der Webpräsenz ihre Dienste auch im WAP-Portal an. Damit stehen den Nutzern die Seiteninhalte in grafisch- sowie inhaltlich stark vereinfachter Form auch mobil zur Verfügung.

2.4.4. Softwaretool für Mobiltelefone

Eine weitere Möglichkeit zur Darstellung der Informationen auf dem Mobilfunkgerät bietet die Erstellung eines Softwaretools. Denkbar wäre hierbei, ein Programm für das mobile Gerät zu entwickeln, welches die Stundenplaninhalte auf dem Display darstellt.
Der Quellcode des Programms wird einmalig, über eine verfügbare Schnittstelle, zum Beispiel Bluetooth, Infrarot oder WAP auf das jeweilige Mobiltelefon geladen und kann von dort aus verwendet werden. Lediglich die Daten zur Aktualisierung werden dann regelmäßig übermittelt. Dies kann über das mobile Internet oder auch durch SMS-Abfrage realisiert werden, wobei jedoch zusätzliche Kosten entstehen. Die Datenübertragung mittels Infrarot, Bluetooth oder einer seriellen Schnittstelle erweist sich dabei als eher unvorteilhaft. Die

Informationen müssten zur Abfrage zentral bereitgestellt werden und wären somit nicht jederzeit mobil verfügbar.
Zur Entwicklung eines solchen Programms eignet sich die von Sun Microsystems eigens für Mobilgeräte entwickelte Programmiersprache Java ME (Micro Edition). Diese ist laut Hersteller [5] bereits auf über einer Milliarde Endgeräte lauffähig.
Da die ständige Aktualisierung der Inhalte über WAP oder SMS notwendig ist, eignet sich eine solche Applikation als Erweiterung der WAP- und der SMS-Lösung (vgl. 2.4.2 – 3).

2.4.5. Einsatz

Viele Unternehmen und Dienstleister nutzen die Möglichkeiten der Informationsverbreitung über das Mobiltelefon um Serviceleistungen, Informationen aller Art und Produkte anzubieten. Die Weiterentwicklung der Technologien ermöglicht dabei immer mehr Wege zur Informationsverteilung.
Einige Beispiele für Firmen, die neben der Internetpräsenz auch eine mobile Erreichbarkeit sicherstellen:

DB Konzern [6] , Online-Reiseauskunft und –Buchung, Erreichbar über:

- http://www.bahn.de/ Internet
- http://mobile.bahn.de/ WAP - Lösung
- Sprachdialog über Telefonhotline

Google Inc. [7], Onlinesuchmaschine, Erreichbar über:

- http://www.google.com/ Internet
- http://wap.google.com/ WAP - Lösung
- Google-SMS Dienst als Branchenbuch

Ebay [8], Onlineauktionshaus, Erreichbar über:

- http://www.ebay.de/ Internet
- http://mobil.ebay.de/ WAP – Lösung
- SMS-Service zur Gebotsabgabe

3. Planung und Entwicklung

3.1. Auswahl des Verfahrens

Die Auswahl des Verfahrens erfolgt hinsichtlich verschiedener Aspekte. Es ist eine Schnittstelle anzustreben, welche vom größten Anteil der potentiellen Nutzergruppe, in diesem Fall die Studierenden, erreichbar ist. Dabei müssen auch Überlegungen einbezogen werden, hinsichtlich Performance, Oberflächengestaltung, Nutzerverhalten, Nutzeransprüche sowie Stand der Technik, Erweiterbarkeit und Wirtschaftlichkeit. Die Umsetzung ist dabei einen Kompromiss aus dem eigentlichen Zweck, der Nutzung und dem erforderlichen Aufwand.
Um eine Entscheidung für das Verfahren, welches am besten geeignet ist, zu treffen, müssen die Möglichkeiten (vgl. 2.4.) objektiv beurteilt werden. Die Tabelle 1 gibt dazu einen Überblick über die jeweiligen Vor- und Nachteile der einzelnen Verfahren. Dabei wurde die Möglichkeit zur Erstellung einer Softwareapplikation für das Endgerät außer Acht gelassen, da diese grundlegend auf den anderen Systemen basiert und die dabei auftretenden Abweichungen letztlich nur in der Methode der Darstellung liegen.

Die Gegenüberstellung der einzelnen Verfahren zeigt, dass jede Methode andere Vor- und Nachteile hat. Zwar bieten das Sprachdialogsystem und der SMS-Service dabei die beste Erreichbarkeit, haben aber beide Nachteile in den Bereichen der Navigation, der aufwendigen Umsetzung und den auftretenden Kosten für Anbieter und Nutzer.
Die WAP-Lösung hingegen bietet einer kleineren Nutzergruppe die Zugriffsmöglichkeit. Die Vorteile hierbei sind die Navigation, die Übersichtlichkeit und die geringeren Kosten. Außerdem kann, je nach Anforderung und Umsetzung, direkt auf die Daten der Webinhalte zugegriffen werden, was wiederum die Umsetzung erleichtert.

Im Hinblick auf die wachsende Nachfrage nach dem mobilen Internet und den aufgezeigten Vorteilen, ist die prototypische Umsetzung an die WAP-Lösung angelehnt und wird in den folgenden Kapiteln weiter behandelt.

Kriterium	**Sprachdialog**	**SMS-Service**	**WAP**
Erreichbarkeit	- über jeden Telefonanschluss	- Standard bei Mobiltelefonen - Festnetzanschlüsse mit SMS-Funktion	- mobile Endgeräte mit Internetfunktion
Bedienung/ Nutzer-freundlichkeit	- durch Anruf - Auftreten langer Dialoge - Verständigungsprobleme - keine Datensicherung	- durch SMS-Versand - Ergebnis abhängig von Formulierung - Datensicherung im Speicher des Geräts	- schnelle und direkte Erreichbarkeit - Datensicherung möglich
Benutzer-oberfläche	- keine graphische Oberfläche - Navigation über Sprache	- keine graphische Oberfläche - Navigation bedingt über SMS-Rückfrage	- graphische Oberfläche - Navigation über Textein-/ausgabe & Formularfelder
Daten-aufkommen - Nutzer -	- keine physischen Daten, da Sprachausgabe	- Umfang einer oder mehrere SMS im Speicher	- Datenverkehr durch Aufruf der Skripte und Darstellung im Browser - benötigt genügend Arbeitsspeicher
Daten-aufkommen - Anbieter -	- Quellcode in VXML - Umfangreiche Skripte für Steuerung da viele Annahmen zu treffen sind - Daten können von Web-präsenz übernommen werden	- Quellcode in HTML - Umfangreiche Skripte für Steuerung da viele Annahmen zu treffen sind - Daten können von Webpräsenz über-nommen werden	- Quellcode für Seitengestaltung - Daten können von Webpräsenz über-nommen werden
Umsetzung	- VoiceXML - sehr umfangreicher Quellcode	- WAP- oder HTML-Seite	- mit XHTML Basic oder WML
Kosten - Nutzer -	- Servicerufnummer - erhöhte Gesprächskosten - je nach Telefonanbieter	- Servicerufnummer - Preise höher als Standard-SMS - je nach Dienstanbieter	- Kosten für Internet-verbindung - Abrechnung nach Zeit o. Datenvolumen - je nach Anbieter
Kosten - Anbieter -	- Nutzung des VoIP-Gateways bzw. VoiceXML Servers - Sonderrufnummer	- für Nutzung des SMS-Servers - Sonderrufnummer	- für Domain
Technische Voraussetzung	- VoiceXML Server - VoiceXML-Interpreter - Gateway	- SMS- Server - Gateway	- Webserver - Gateway

Tabelle 1: Gegenüberstellung der Verfahren

3.2. Funktionalität

Um ein System zu entwickeln, ist es zunächst notwendig zu Klären, was das System leisten soll. Den Studenten, Dozenten und Mitarbeitern steht, mit der Einführung eines hochschulweiten Onlinestundenplans, ein Informationssystem im Internet zur Verfügung, welches Aufschluss über Lehrveranstaltungen, Termine sowie Raumbelegungen gibt. Ferner können Informationen, wie etwa Angaben über Dozenten und Modulbeschreibungen zusätzlich abgefragt werden und verschiedene Suchfilter bieten individuelle Übersichten.
Die mobile Anbindung soll kein ebenbürtiger Ersatz für das Onlinesystem werden. Die Grundidee ist es, den Studenten eine möglichst einfache Informationsquelle zur kurzfristigen Abfrage der anstehenden Lehrveranstaltungen zu bieten. Auch im Zusammenhang mit den Ladezeiten und den damit verbundenen Onlinenutzungskosten, sowie der begrenzten Anzeigekapazität sollte der Datenverkehr so gering wie möglich gehalten werden.
Das betrifft nicht nur die endgültig angezeigten Informationen, sondern auch das Laden der Skripte zur Seitendarstellung, die Verarbeitung der Anfrage, die Größe der gesendeten Daten sowie die Verarbeitung der Antwort.

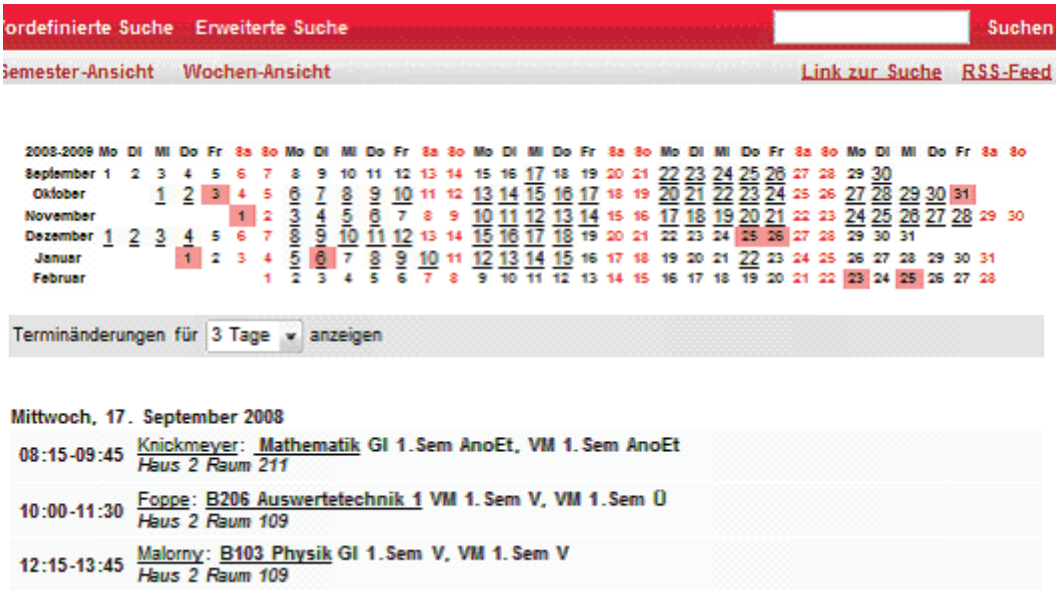


Abbildung 4: Testversion des Onlinestundenplan HS-NB

Die wichtigsten Informationen im Stundenplan sind neben der Bezeichnung der Lehrveranstaltung die Zeit, die Art und der Veranstaltungsort, welcher aus Gebäude- und Raumnummer besteht. Allerdings gibt es auch Veranstaltungen, die aus mehreren Teilen bestehen und dadurch von verschiedenen Dozenten durchgeführt werden. Um kenntlich zu machen, um welche Veranstaltung es sich dabei handelt, ist es sinnvoll zusätzlich noch den Dozenten mit auszugeben.
Damit die Informationen auf den kleinen Displays möglichst lesbar und übersichtlich dargestellt werden können, sind diese kurz zu halten. So empfiehlt es sich, anstelle der langen Bezeichnungen für die einzelnen Veranstaltungen, auf die Modulnummer zurückzugreifen. Diese kann darüber hinaus durch eine Verlinkung auf einen Glossar verweisen in dem die dazugehörige Beschreibung zu finden ist. Analog kann auch mit dem Dozenten verfahren werden, wobei hier dann ein Namenskürzel angezeigt wird.

Als mögliches Ausgabebeispiel:

Zeit	Kurs	Art	Ort	Dozent
08:15-09:45	B123	V	1 / 234	Prof. …
10:00-11:30	B345	Ü	1 / 567	Prof. …

Tabelle 2: Beispiel für Ausgabe

3.3. Softwaremittel

Die für Entwürfe, Testzwecke und Umsetzung eingesetzten Softwaremittel richten sich nach der, für den Aufbau des Stundenplans verwendeten Programmiersprache PHP in Verbindung mit der MySql-Datenbank und der Auszeichnungssprache XHTML Basic für Kleinstinformationsgeräte als eingeführter Standard des W3C.
Da für Tests ein Webserver benötigt wird, eignet sich der XAMPP als Zusammenstellung von Open-Source-Software. Dieser beinhaltet einen Apache Webserver, eine MySql-Datenbank, sowie die Skriptsprachen PHP und Perl.

Software	Version	Verwendung
XAMPP	1.7.0	Open-Source-Softwarepaket
- Apache	2.2.11	Webserver
- MySql	5.1.30	Datenbank
- PHP	5.2.8 +	Skriptsprache
- phpMyAdmin	3.1.1	GUI für MySql Datenbank
Maguma Open Studio	1.0-PR2	PHP-Editor
XHTML-Basic	1.0	Auszeichnungssprache
MS Windows	6.0	Betriebssystem

Tabelle 3: Verwendete Softwaremittel

3.4. Benutzeroberfläche

Die Benutzeroberfläche muss klar strukturiert, lesbar, übersichtlich, eindeutig und leicht bedienbar sein. Es ist zu vermeiden lange Texteingaben vom Bediener zu fordern, da diese im Normalfall über die 10 Wähltasten des Mobilfunkgerätes oder Touchscreens erfolgt. Auch fehlt die Unterstützung der mittlerweile weit verbreiteten T9-Wörterbücher, da Eigennamen, wie zum Beispiel die Namen der Dozenten, spezielle Kursbezeichnungen oder Fachbegriffe nur gering unterstützt werden.

Die Abbildung 5 zeigt einen einfachen, in HTML geschriebenen Entwurf für eine mögliche Darstellung und Bedienung durch das Mobiltelefon. Dahinter liegt eine zweckmäßige Datenbank, welche die nötigen Informationen bereitstellt.

Die Navigation erfolgt über Schaltflächen und grenzt somit Schritt für Schritt die Auswahlmöglichkeiten ein. Zuerst der Fachbereich, danach eine Auflistung aller dazugehörigen Studiengänge und im Anschluss die für den Studiengang verfügbaren Semester. Lediglich für das Datum sind Textfelder vorgesehen. Allerdings kann hier das aktuelle Datum als Voreinstellung (default) stehen.

Dieser HTML-Entwurf wurde für die Recherche, Analyse und für Testzwecke entwickelt und dient ausschließlich der Veranschaulichung.

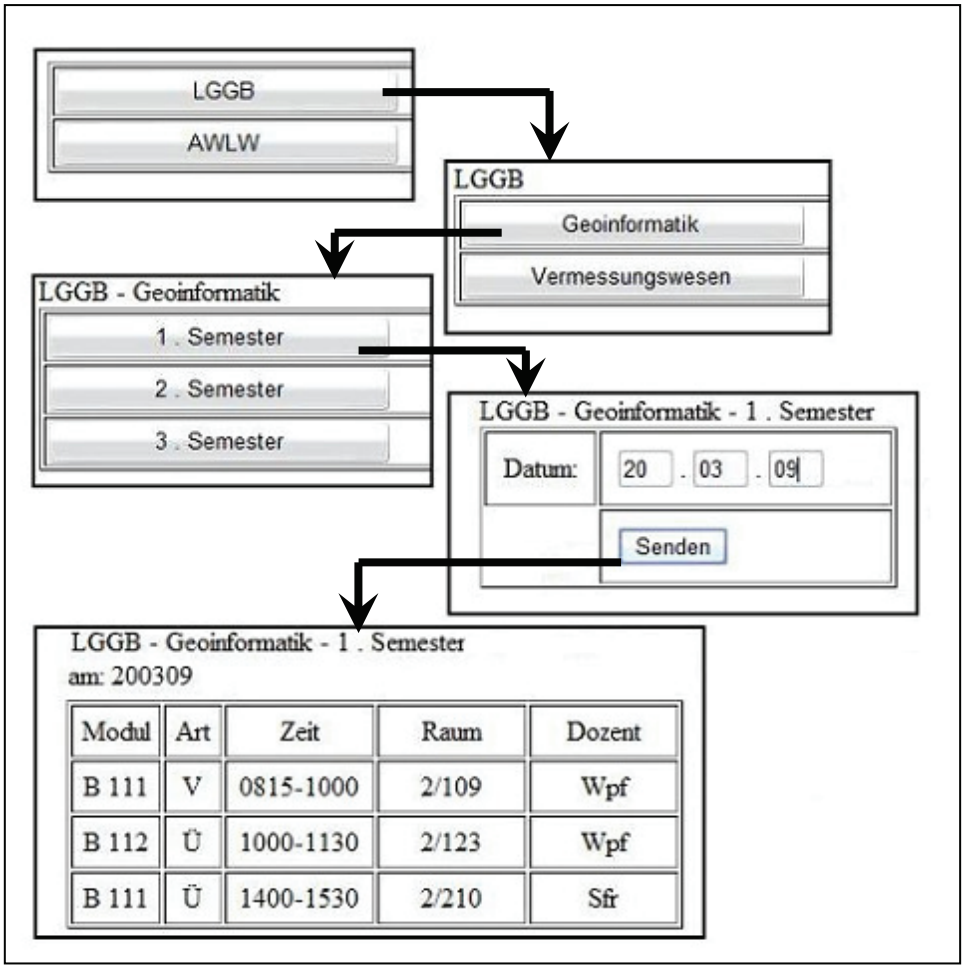


Abbildung 5: GUI Entwurf

4. Lösungsansatz

Für den Entwurf zur Umsetzung bieten sich verschiedene Verfahren an. Diese unterscheiden sich vor allem im Ablauf und der Art der Abfrage sowie dem Datenmanagement.

4.1. Zugriff über Schnittstelle

Die Idee dieses Entwurfs ist es, eine eigenständige und in sich geschlossene WAP-Seite zu erstellen, welche unabhängig vom Quelltext und dem Datenbestand der WWW-Seite läuft und somit an verschiedene Systeme andocken kann.
Das Problem hierbei ist, eine Schnittstelle zu definieren, welche ohne großen Aufwand die benötigten Daten aus dem bestehenden System abfragt und in geeigneter Form bereitstellt. Zwar kann davon ausgegangen werden, dass in verschiedenen Stundenplansystemen grundlegend die gleichen Daten verwaltet werden, aber die Vielzahl der unterschiedlichen Informationen und die Unterschiede in Struktur und Aufbau machen die Gestaltung einer solchen Schnittstelle aufwendig.

Als Informationsquelle besteht die Möglichkeit, ein eigenständiges Datenbanksystem anzulegen, welche die für die Ausgabe benötigten Informationen enthält. Die Daten können dabei manuell in diese Datenbank eingegeben werden, womit das System komplett unabhängig wäre. Allerdings ist das sehr zeitaufwendig und darüber hinaus unnötig, da die Daten bereits in der Datenbank des Onlinestundenplans vorhanden sind.
Hier würde sich eine Schnittstelle anbieten, welche die benötigten Daten aus der Datenbank der Webseite abruft, in die der WAP-Seite integriert und regelmäßig Änderungen abgleicht. Das Format der bereitgestellten Daten richtet sich dabei nach der Umsetzung der mobilen Lösung. Zum einen kann dies, wie bereits erwähnt, in Form eines eigenständigen Datenbanksystems oder aber als Datensatz in einem Textfile geschehen.

4.1.1. Durch eigenständige Datenbank

Die Überlegung, eine eigenständige Datenbank einzurichten, hat den Vorteil, dass sich die Ausgabe weitestgehend flexibel gestalten lässt und nicht an eine feste Form, wie etwa ein Textfile, gebunden ist.

Die Seite wird über den Browser des Mobiltelefons aufgerufen und fordert zunächst die Eingabe der benötigten Suchparameter: *Fachbereich, Studiengang, Semester, Gruppe* und *Datum*.

Anhand der Eingabe wird mittels SQL die Datenbankanfrage generiert und gesendet. Das Ergebnis kann dann ausgewertet und ausgegeben werden (Abbildung 6).

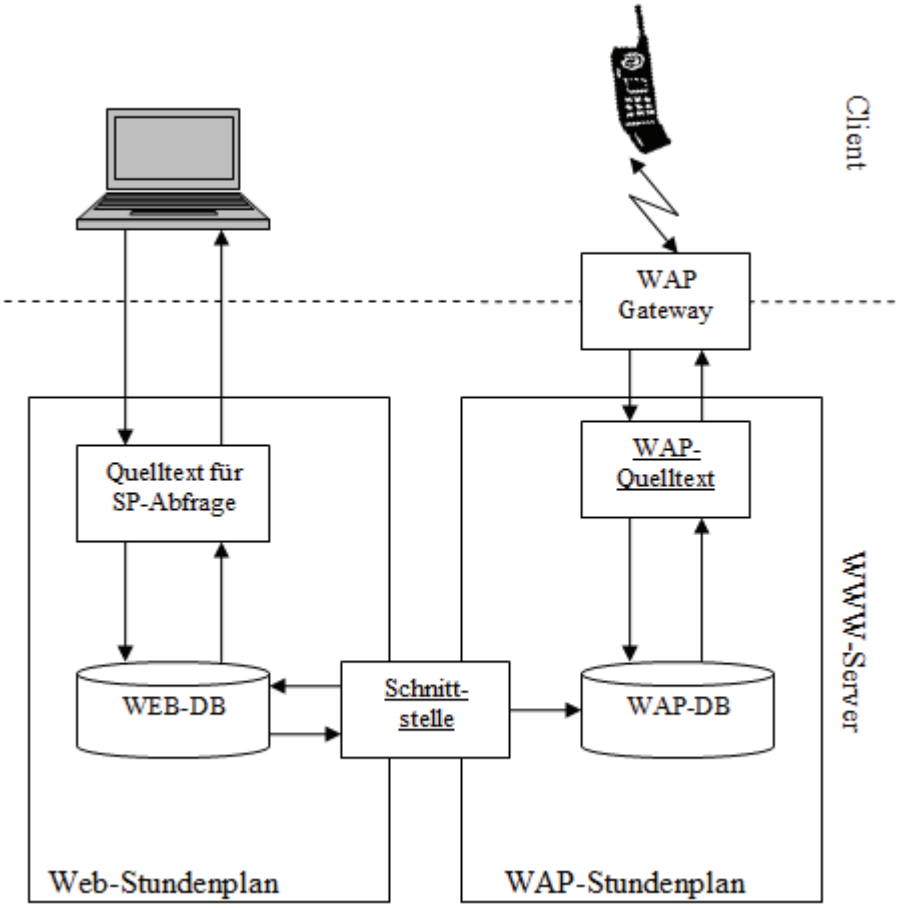


Abbildung 6: Darstellung Datenbankbasiertes Konzept

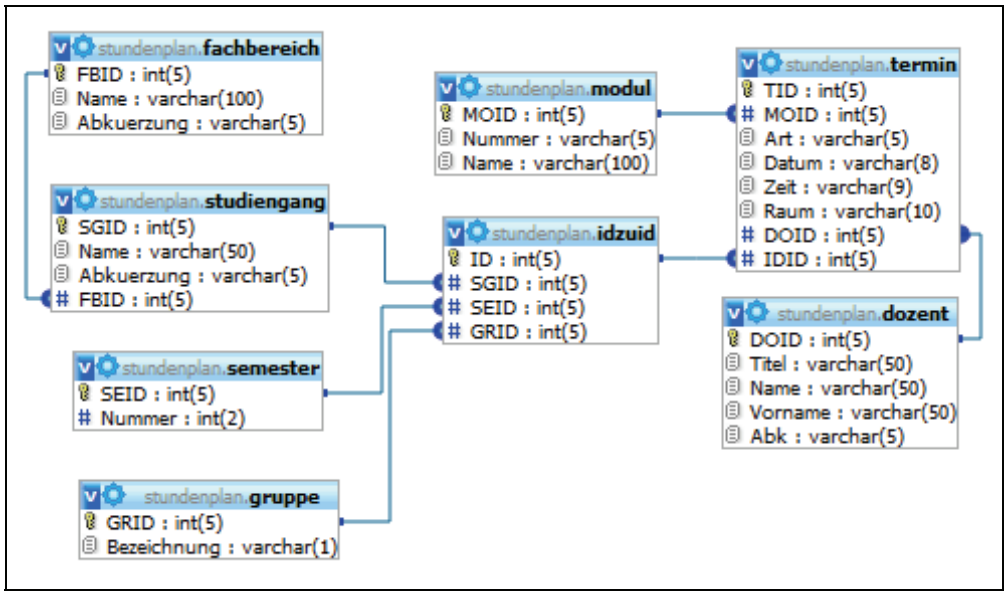


Abbildung 7: DB-Entwurf für datenbankbasierte Lösung

Um ein funktionsfähiges Datenbankmodell zu erstellen, muss in der Vorbetrachtung festgelegt werden, welche Daten nötig sind und für welche Zwecke diese gebraucht werden. In der Abbildung 7 ist eine Möglichkeit für den Aufbau der Datenbank dargestellt.
Anhand des Datenbankmodells kann dann die Schnittstelle zwischen den beiden Systemen definiert werden. Diese hat die Aufgabe die benötigten Daten aus der Web-Datenbank auszulesen und in die WAP-Datenbank zu integrieren. Die WAP-Anfrage hat darauf keinen Einfluss. Der Zeitpunkt der Aktualisierung wird von der Schnittstelle aus gesteuert und kann unmittelbar nach auftreten von Änderungen oder in einem definierten Rhythmus (12-stündig, täglich, wöchentlich) erfolgen. Die Funktion dieser Schnittstelle ist im Abschnitt 4.1.3. beschrieben.

In der Abbildung 8 ist in einem Programmablaufplan ein möglicher Ablauf für eine Anfrage dargestellt. Nach Aufruf der WAP-URL im Browser des Mobiltelefons, wird zunächst die Seite für die Eingabe der Suchkriterien angezeigt. Werden diese korrekt eingegeben, wird die Datenbankanfrage gestartet und das Ergebnis ausgewertet.

Die Darstellung erfolgt dann im Browser des Handys.

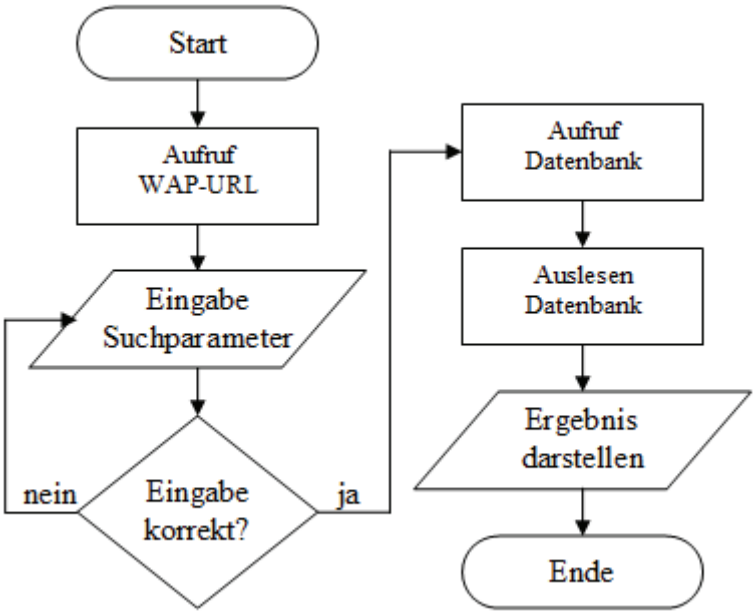


Abbildung 8: PAP –Entwurf für Datenbank- Lösung

4.1.2. Basierend auf Textfiles

Dabei wird die WAP-Seite so angelegt, dass das für die Auswertung und Ausgabe zuständige Skript auf ein, durch eine Schnittstelle erstelltes, inhaltlich vordefiniertes Textdokument zugreift. Ein so genannter Parser sorgt dann dafür, dass der Inhalt des Dokuments in die benötigten Bestandteile zerlegt und für die Ausgabe in das geeignete Format umgewandelt wird.
Um sicher zu stellen, dass jeder Zugreifende tatsächlich die Informationen bekommt, die er auch benötigt, müssen weitere Annahmen getroffen werden. Daher sollte bereits vor dem Auslesen der Inhalte aus dem Textfile klar sein, welchen Studiengang, welches Semester, welche Gruppe und welches Datum die Anfrage betrifft. Dies kann beim Aufrufen der Anwendung über Formularelemente erfolgen (*vgl. 3.4 Benutzeroberfläche).*

Es empfiehlt sich hierbei eine Datenbank zu integrieren, in der die Informationen für die Vorauswahl verwaltet werden. Der Aufbau kann gemäß Abbildung 9 realisiert sein. Der Bezug der Daten erfolgt dabei entweder manuell oder angelehnt an das Konzept mit der eigenständigen Datenbank.

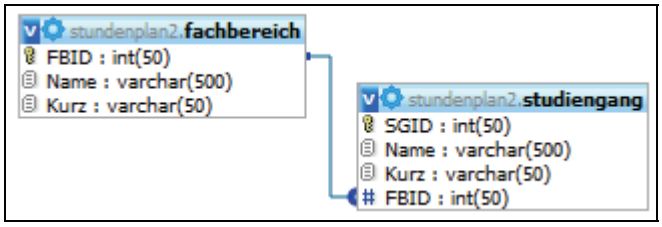


Abbildung 9: Datenbankaufbau für Textfile-Lösung

Im Textfile sollen alle Daten enthalten sein, welche für die Auswertung und die Ausgabe relevant sind. Hierbei kommt es schon im Vorfeld darauf an, die Daten logisch zu strukturieren um spätere Fehler und Unstimmigkeiten bei der Auswertung zu vermeiden. Auch muss darauf geachtet werden, dass sich der Inhalt eines einzelnen Files aus den Stundenplänen aller Studiengänge für das gesamte laufende Semester zusammensetzt und dadurch schnell sehr groß werden kann. Da bei der Auswertung das ganze File geladen wird, wirkt sich das negativ auf die Ladezeit im WAP-Browser des Mobiltelefons aus.
Daher ist es sinnvoll mehrere kleine Files zu erstellen, so dass der Browser gezielt auf einen Datensatz zugreifen kann. Die Trennung kann über den Fachbereich, oder auch den Studiengang durchgeführt werden. Je nach Trennungskriterium bekommt dann das jeweilige Textfile den Namen des Fachbereichs oder des Studiengangs, als Beispiel: *LGGB.txt* oder auch *GI.txt, LT.txt.*

Erzeugt und aktualisiert wird das Textfile über die definierte Schnittstelle (4.1.3 Beschreibung der Schnittstelle). Dies geschieht, ohne Einfluss der WAP-Anfrage, automatisch. Sinnvoll ist dabei einmal am Tag die Aktualisierung durchzuführen, um Änderungen zeitnah in die WAP-Übersicht zu integrieren. Der Ablauf ist in Abbildung 10 dargestellt.

Das Format wird durch den für die Auswertung zuständigen Programmteil definiert und muss beim Erstellen des Quelltextes für die Schnittstelle zwischen Datenbank und Textfile beachtet werden.

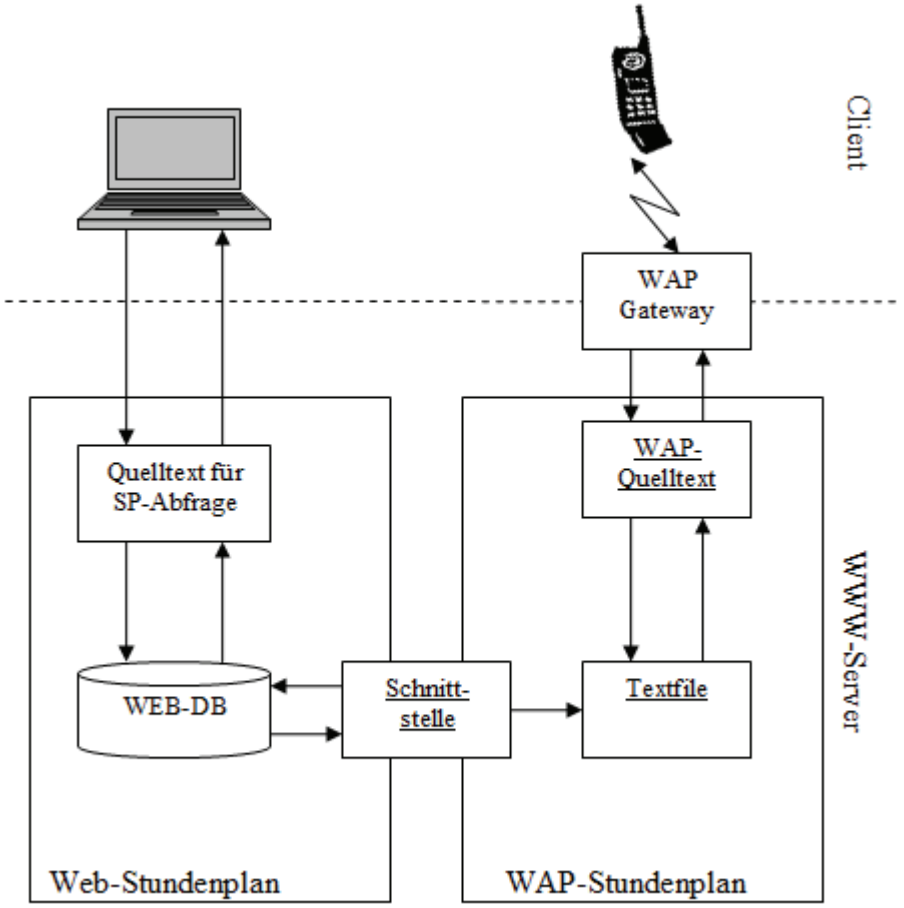


Abbildung 10: Darstellung des textfilebasierten Konzepts

Die Abbildung 11 zeigt einen Entwurf für den möglichen Aufbau eines Textfiles. Es handelt sich dabei um einen Auszug aus einem File, welches die Stundenplandaten aller Studiengänge des Fachbereichs LGGB beinhaltet. Das File lässt sich in Blöcke unterteilen, welche wiederum aus Blöcken bestehen können. Jeder Block beginnt mit einer „#“, gefolgt von einer Kennung („SG=“, „Sem=“, „Grp=“) mit einer Spezifikation („GI“, „1“, „A“) und endet mit der Zeile vor dem nächsten Auftreten der Kennung.

```
LGGB.txt

#SG=GI,
#Sem=1
#Grp=A
#Date=15.06.2009
08:15-09:30; B123; V; 1/234; Prof
10:00-11:30; B345; Ü; 1/567; Prof
#Date=06.06.2009
08:15-09:30; B123; Ü; 1/234; Prof
10:00-11:30; B345; V; 1/567; Prof
#Grp=B
...
#Sem=  2
#Grp=A
...
#SG=VM
...
EOF
```

Abbildung 11: Möglicher Aufbau Textfile

Der für die Auswertung des Textfiles verantwortliche Programmteil (für die bessere Verständlichkeit als Parser bezeichnet), zerlegt das Dokument in seine Bestandteile und gibt die benötigten Informationen weiter zur Ausgabe.
Für die Zerlegung wird das Dokument zeilenweise ausgelesen. Dabei wird der jeweilige String mit einem definierten Suchstring oder auch einem regulären Ausdruck verglichen. Findet eine Übereinstimmung mit dem Suchmuster statt, folgt gegebenenfalls die Suche nach einem weiteren Kriterium oder der gesuchte String wurde gefunden und kann weiter verarbeitet werden. Für die Suche bieten sich die PHP-Befehle „strpos()“ für einen Suchstring und „preg_match()“ für reguläre Ausdrücke an.
Das #-Zeichen am Zeilenanfang dient dabei der Vereinfachung der Suche nach den jeweiligen Zeichenketten und zeigt gleichzeitig dem Betrachter, dass es sich in dem Fall um einen String handelt, der zur Eingrenzung der Ergebnisse dienlich ist.

Beispiel für einen Ablauf:
Die Anfrage lautet nach LGGB, Geoinformatik, 1. Semester, Gruppe A am 06.06.2009:
Das Textfile *LGGB.txt* wird gesucht, geöffnet und ausgelesen.
Mit *„strpos ($String, ‚#SG=GI’);“* wird nach dem Studiengang gesucht, wobei „GI“ aus der Datenbank entnommen werden kann. Bis zum nächsten Auftreten des Zeilenanfangs *„#SG=…“* gehören alle weiteren Einträge zum Studiengang. Analog verläuft dann die Suche nach Semester, Gruppe und Datum.
“preg_match(“([0-2][0-9]:[0-5][0-9])”, $String, $Treffer, [Flag], [Offset]);” überprüft dann zeilenweise nach dem nächsten Auftreten des Uhrzeitformats *hh:mm*. Hierbei wird eine Schleife benötigt, da die Suche beim ersten Auftreten endet. Das Schleifenende wäre dann das nächste auftreten des Symbol „#“. Beim setzten des optionalen Parameters „Flag“ wird zusätzlich die Position des ersten Auftretens mit ausgegeben, der Parameter „Offset“ bewirkt die Suche ab dieser Position.
Mit „*explode(";", $String)*“ wird die Zeichenkette anhand des Semikolon aufgetrennt und in einem Array abgelegt. Dieses kann dann zur Ausgabe weitergegeben werden.

Dieses Konzept birgt allerdings einen großen Nachteil. Durch die Umsetzung mit dem definierten Textfile beschränkt sich die Suchanfrage auf den Inhalt dieses Dokuments. Dadurch lassen sich spätere Erweiterungen, wie etwa eine Filteroption für die Dozentensuche, nur mit größerem Aufwand integrieren.

Der Programmablaufplan aus Abbildung 12 stellt den möglichen Ablauf einer Anfrage dar. Zunächst wird die WAP-URL im Browser aufgerufen. Nach Eingabe und Prüfung der Suchkriterien erfolgt dann das Einlesen des Textfiles.
Die Daten werden im Parser (Abbildung 13) ausgewertet und im Browser des Mobilfunkgerätes dargestellt.

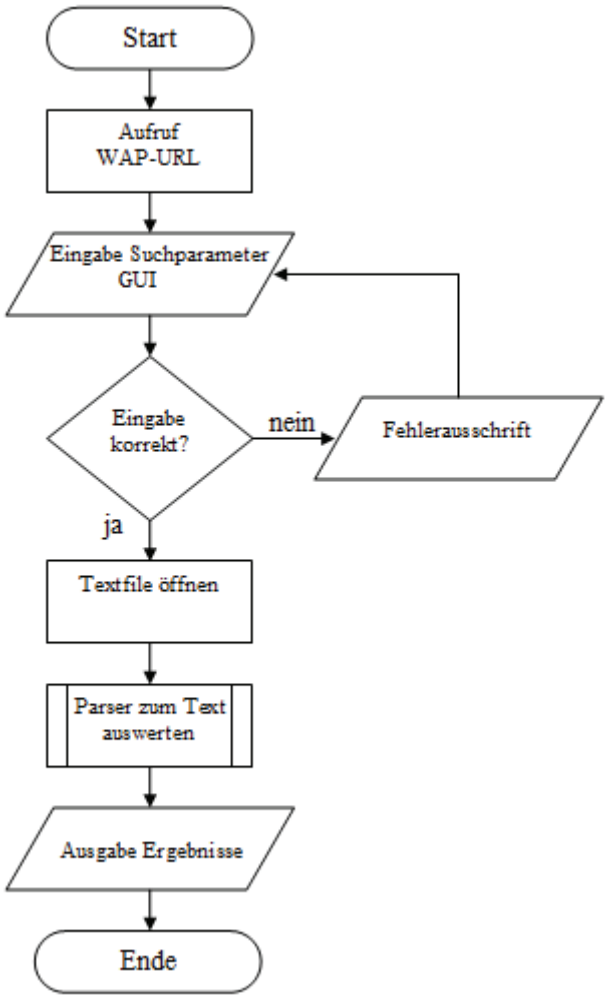


Abbildung 12: PAP-Entwurf für Textfile-Lösung

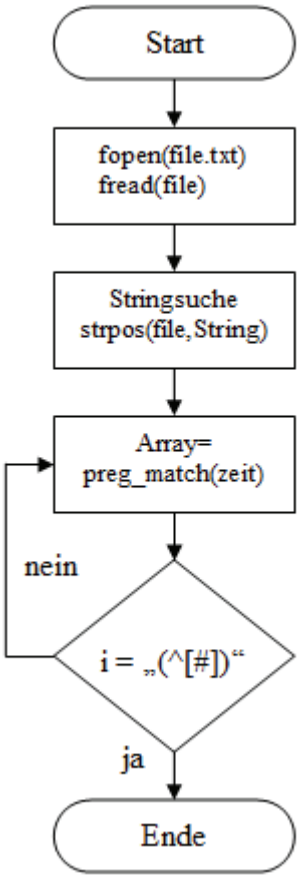


Abbildung 13: PAP-Entwurf Parser

4.1.3. Beschreibung der Schnittstelle

Die Schnittstelle zwischen dem Web- und dem WAP- System hat die Aufgabe, die benötigten Daten und Informationen aus der Web-Datenbank abzufragen, in die notwendige Form zu überführen und für den WAP-Zugriff bereitzustellen. Die Schnittstelle kann für die Übersichtlichkeit in 4 Abschnitte aufgegliedert werden (Abbildung 14).

Programmabschnitt I regelt dabei die Abfrage der Informationen von der Web-Datenbank, Programmabschnitt II bereitet diese Daten für die Weitergabe vor, welche vom Abschnitt III in das WAP-System übergeben werden. Der Abschnitt IV ist zuständig für die Konfiguration, den Programmablauf und der Definition von Variablen und Funktionen.

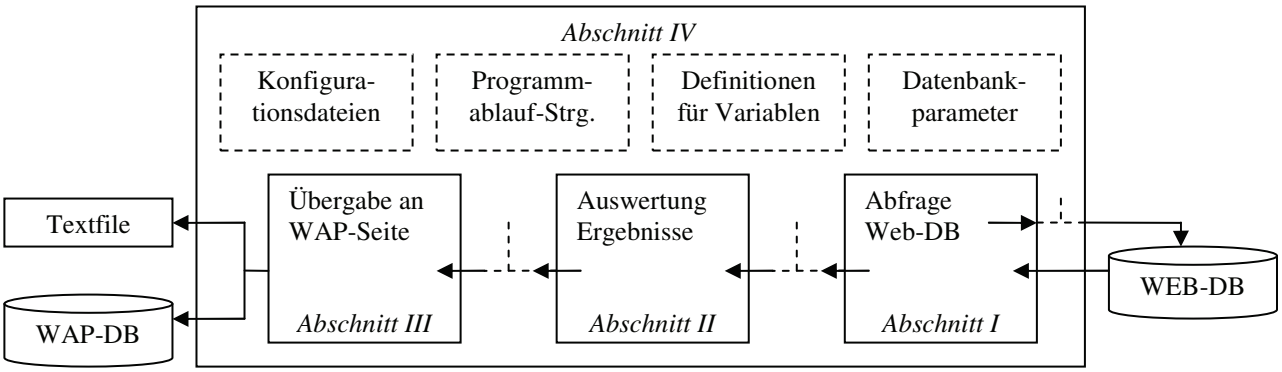


Abbildung 14: Darstellung der Schnittstelle

Der Programmabschnitt I behandelt die Abfrage der Informationen aus der Datenbank der Webseite. Die Abfragen richten sich dabei nach der Datenstruktur der Quelle. Da die Schnittstelle universell gehalten werden soll, um auch andere Stundenplansysteme verwenden zu können ist hier besondere Sorgfalt in der Planung und Umsetzung nötig, damit im Falle einer Systemumstellung der Aufwand minimal bleibt.
Die Schnittstelle wird so angelegt, dass nur in diesem Abschnitt I Änderungen bei einer Umstellung nötig sind. Der Gedanke dabei ist, dass unterschiedliche Systeme zwar eine unterschiedliche Struktur aufweisen, sich die eigentlichen Daten aber weitestgehend ähneln. Hierbei wird vorausgesetzt, dass jedes Informationssystem für Lehrveranstaltungen neben der Bezeichnung mindestens über eine Zeit-, Orts- und Personenkomponente verfügt, eben jene Informationen, die auch für eine aussagekräftige WAP-Darstellung hinreichend sind.

Die Unterschiede belaufen sich dabei auf den Aufbau der Datenbanken, wie etwa bei der Organisation der Daten, den Beziehungen und den Bezeichnungen der Spalten, Tabellen und der Datenbanken.

```
Funktion Abfrage() {
     SQL = SELECT … FROM … WHERE …
     Datenbank aufrufen
          mysql_connect(Server, Nutzer, Passwort)
          mysql_select_db (DB-Name, Resource)
     Ergebnis = mysql_query (SQL)
     Rückgabe return Ergebnis    }
```

Die SQL-Statements können hier, je nach Stundenplansystem, an die jeweilige Datenbank angeglichen werden.
Im Quelltext der gesamten Schnittstelle kann dieser Abschnitt als Funktion angesehen werden, welche ohne Übergabeparameter aufgerufen wird, die benötigten SQL-Statements an die Datenbank übermittelt und das Ergebnis-Handle zur weiteren Auswertung an den Programmabschnitt II zurückgibt.

Der Programmabschnitt II wertet dann das Ergebnis-Handle aus und legt die Resultate in einem Array ab. Dieses Array besitzt eine definierte Struktur (Tabelle 4).

	[0] *(FB)*	**[1]** *(SG)*	**[2]** *(Sem)*	**[3]** *(Grp)*	**[4]** *(Date)*	**[5]** *(Zeit)*	**[…]** …
[0]	LGGB	GI	1	A	05.06.09	0815-0930	…
[1]	…						
[2]							
[…]							
[n]							

Tabelle 4: Aufbau Array

Mit Hilfe von Schleifen, Bedingungen und den passenden PHP-Befehl kann das Ergebnis ausgelesen und in dem Array abgelegt werden, dies ist im Folgenden mit der Funktion *mysql_result()* dargestellt.

```
Rz = mysql_num_rows(Ergebnis)
Cz = mysql_num_fields(Ergebnis)

for (i=0;i<Rz;i++){
     for (k=0;k<Cz;k++){
          Array[i][k] =mysql_result(Ergebnis, i, k) }}
```

Ist das Array angelegt und im richtigen Format, kann die Übergabe beziehungsweise die Bereitstellung der Daten für das WAP-System im Abschnitt III erfolgen. Dieser hat festgelegten Quelltext und braucht bei Änderungen der Systeme nicht verändert werden.
Der Aufbau und die Funktionsweise dieses Abschnittes richten sich nach dem jeweiligen Konzept für die Datengrundlage der WAP-Seite.
Bei dem Textfile als Grundlage wird das Array ausgelesen und zeilenweise, unter Beachtung des vorgegebenen Formats (vgl. 4.1.2, Abbildung 13), in ein Textdokument geschrieben.
Bei einer Datenbank als Grundlage muss das Array zerlegt werden und die Daten können dann über den SQL-Befehl *INSERT … INTO* in die WAP-Datenbank eingepflegt werden.

Um die regelmäßige Aktualisierung der Daten zu gewährleisten, muss die Schnittstelle im festgelegten Rhythmus aufgerufen werden.
Dies kann mit einem CronJob sichergestellt werden.
*00 20 * * 1-5 php /'Pfad'/interface.php* ist der Eintrag in der CronTab, welcher an Werktagen um 20 Uhr die Schnittstelle aufruft.

4.2. Zugriff über URL der Webseite

Dieser Ansatz basiert auf der Überlegung, die Informationen über den Quellcode des bestehenden Onlineplans abzufragen. Dieses vereinfacht das Skript der WAP-Seite enorm, da keine eigenen Abfragen generiert werden müssen und der Umweg über ein eigenes Datenbanksystem erspart bleibt.
Im Quelltext der Webseite sind die nötigen SQL-Statements, Datenbankparameter sowie Optionen für die Abfrage bereits definiert.
Denkbar ist dabei, dass der Nutzer die Anfrage über den WAP-Browser an das WAP-Gateway sendet, dort wird sie in ein HTTP-Verständliches Format umgewandelt und an den Webserver mit der Schnittstelle weitergeleitet. Diese wird aufgerufen und erzeugt eine verständliche HTML-Anfrage für den Quelltext des Web-Stundenplans. Die Anfrage wird ausgewertet und die Datenbankabfrage durchgeführt. Das Ergebnis wird zurück zur Schnittstelle geleitet, wo dann die Auswertung und Darstellung der Daten stattfindet (Abbildung 15).

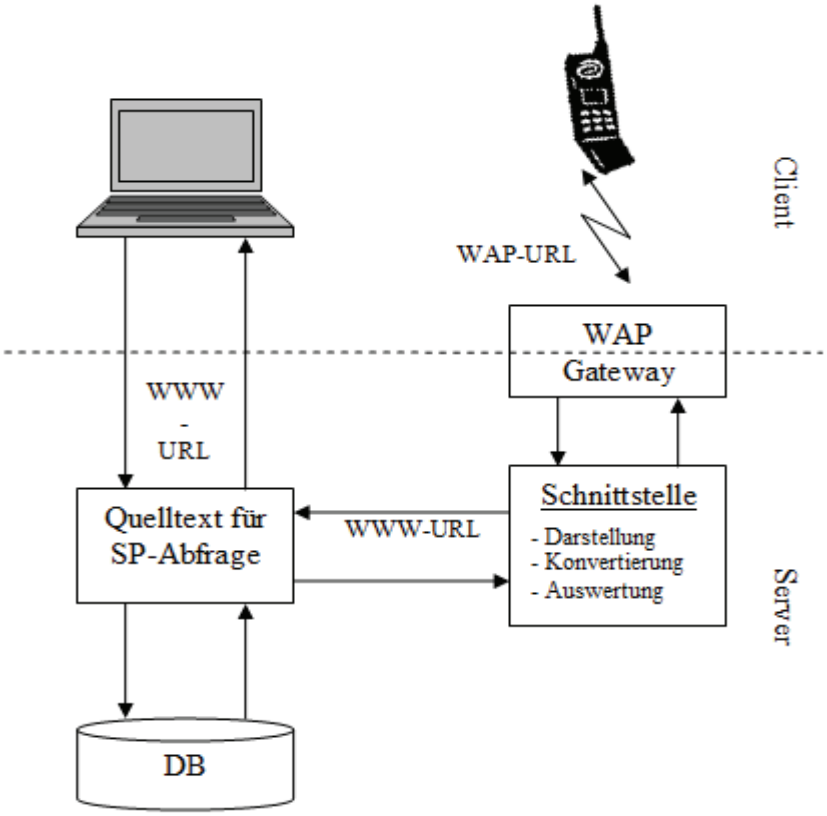


Abbildung 15: Entwurf für Ablauf einer URL-basierten Abfrage

Aufgerufen wird die Webseite über die Web-URL. Dabei werden alle benötigten Übergabeparameter für eine Suchabfrage gleich mit übergeben. Es wird sozusagen ein normaler Zugriff inszeniert. Um das umzusetzen zu können, müssen zunächst die Zusammensetzung der URL sowie die einzelnen Parameter untersucht und ausgewertet werden.

URL mit Parameter für Onlinestundenplan:
http://portal.hs-nb.de/~stundenplan/index.php?org=&jhg=&gru=&sem=&vordefsuche=

Teil der URL	***Funktion***
http://portal.hs-nb.de/~stundenplan/index.php	Protokoll, Host, Pfad
?org=	Übergabevariable für Studiengang
&jhg=	Übergabevariable für Semester
&gru=	Übergabevariable für Gruppe
&sem=	Übergabevariable für aktuelles Semester
&vordefsuche=	Übergabevariable für gesetzten Suchfilter

Tabelle 5: Aufbau URL des Hochschulstundenplans

5. Prototypische Umsetzung

5.1. Grundlagen

Dieses Kapitel beschäftigt sich mit der Umsetzung des Lösungsansatzes 4.2, den Zugriff auf die Webdatenbank über die URL der Webseite und soll die Herangehensweise und die Funktionsfähigkeit verdeutlichen.
Dazu wurde mit der Programmiersprache PHP ein lauffähiger Prototyp erstellt, welcher auf die Testversion des Hochschulstundenplans zugreift und die dort hinterlegten Testdaten als Datengrundlage nutzt.
Das Programm ist so angelegt, dass eine Standardabfrage durchgeführt werden kann, bei der das Ergebnis eine Übersicht über die Lehrveranstaltungen eines gewählten Datums ist. Dabei werden auch der Studiengang und das Semester berücksichtigt.

Die Abbildung 16 zeigt den Programmablaufplan, nach dem die Umsetzung ausgerichtet ist.
Der Programmablauf kann in 4 Teile gegliedert werden. In Eingabe, Erzeugung der URL, Verarbeitung der Ergebnisse und Ausgabe.

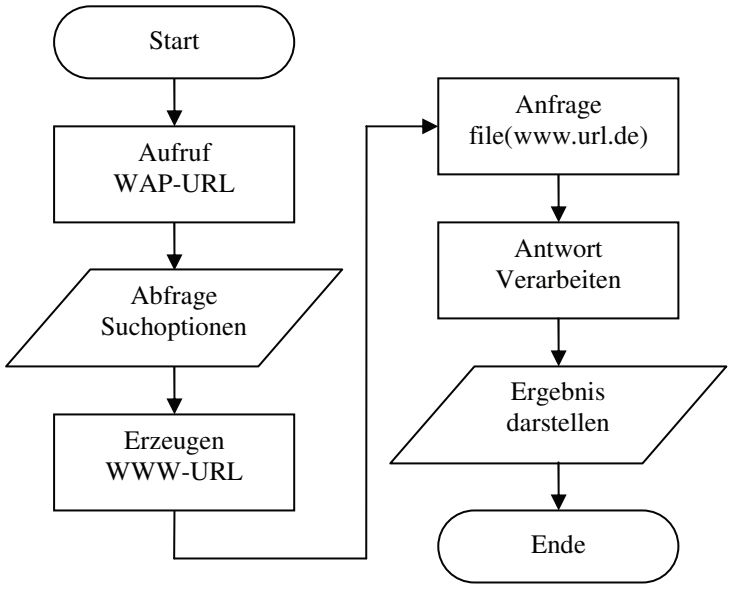


Abbildung 16: PAP Umsetzung

5.2. Eingabe/Ausgabe (GUI)

Die Benutzeroberfläche wurde so angelegt, dass die Übersichtlichkeit vorhanden bleibt, und die Ladezeit so gering wie möglich gehalten wird. Auf unnötige Grafiken und Elemente wird dabei verzichtet und der Aufbau der Seite wird so einfach wie möglich gehalten.

Die Startseite (Abbildung 17) besteht aus einem Begrüßungstext und bietet verschiedene Schaltflächen zur Auswahl von Suchfiltern und anderen Optionen. Allerdings ist derzeit lediglich das Feld *Kurssuche* hinterlegt. Die Schalter *Dozentensuche* und *Hilfe* sind noch ohne Funktion und dienen hier als Illustration.

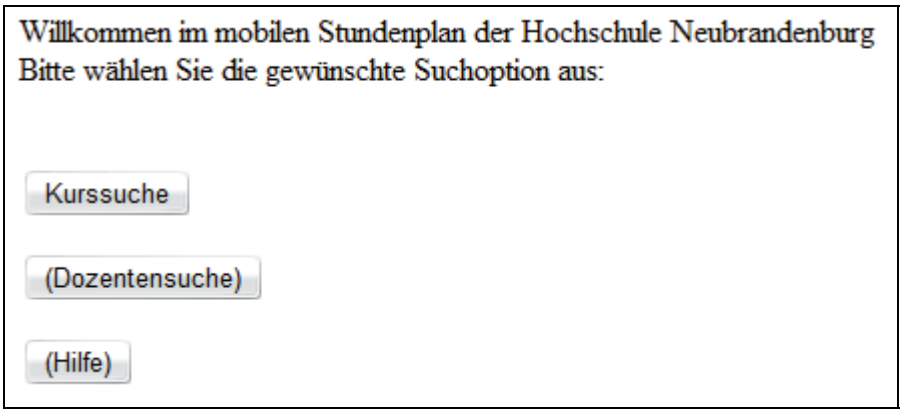


Abbildung 17: Startseite

Nach Aktivierung einer Schaltfläche erfolgt dann der Aufruf des Skripts, welches für die jeweilige Option zuständig ist. In diesem Fall ist es das Skript für die Kurssuche.
Hier wird zunächst die Auswahl des Fachbereichs in einer Select-Box gefordert. Nach erfolgter Auswahl erscheinen dann eine Reihe weitere Formularfelder, welche die Auswahl des Studiengangs, des Semesters, optional der Gruppe und des gewünschten Datums verlangen (Abbildung 18). Die Auswahlliste für den Studiengang wird aus einer Datenbank abgefragt und richtet sich nach dem ausgewählten Fachbereich. Die Auswahlfelder für Studiengang und Gruppe verfügen über statische Werte. Der Gedanke dabei war, die Liste der Studiengänge wegen der Übersicht kurz zu halten und anhand des gewählten Fachbereichs dynamisch zu erstellen. Die Auswahl der Fachsemester hingegen ist auf 8 begrenzt und bleibt dadurch übersichtlich. Außerdem erspart das weitere Abfragen der Datenbank und reduziert somit den Datenfluss.

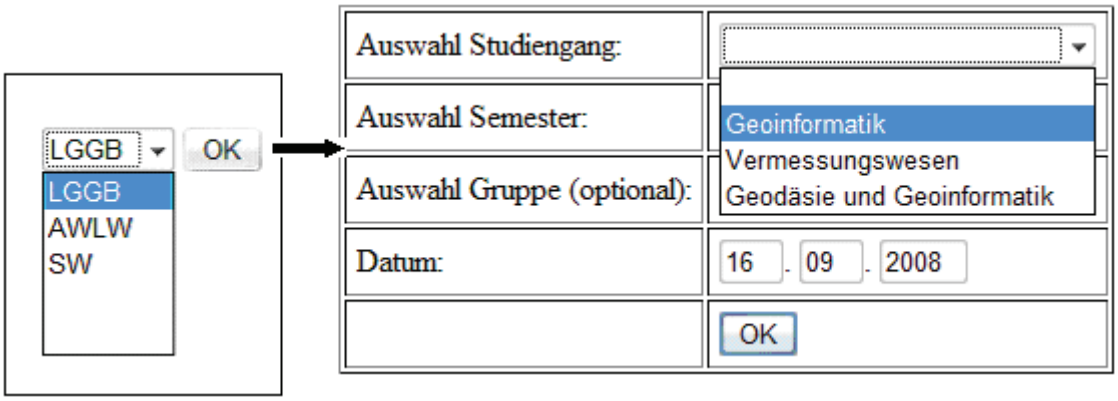


Abbildung 18: Auswahl Fachbereich, Studiengang, Semester und Datum

Das Datum wird über Textfelder im Format TT.MM.JJJJ eingegeben. Es handelt sich hierbei um Ziffern, welche bequem über die Tastatur des Mobiltelefons gewählt werden können. Die Eingabe der trennenden Punkte wird hierbei umgangen. Als Voreinstellung soll hier das aktuelle Datum stehen. Da sich der Testdatensatz auf das Wintersemester 2008 bezieht, ist das Datum vom ersten Eintrag gesetzt.

Die gesetzten Optionen werden dann mittels GET-Methode zur Auswertung (5.3 Verarbeitung der Daten) weitergeleitet. Der Nutzer erhält daraufhin die Antwort, welche in seinem Browser dargestellt wird.

Das Resultat der Anfrage Geoinformatik, 1.Semester am 16.09.2008 ist in der Abbildung 19 dargestellt.

16. September 2008

Zeit	Dozent	Kurs	Art	Ort
08:15-09:45	Hillmann	B102	V	2 / 109
12:30-14:00	Kresse	B105	V	2 / 211
14:15-15:45	Hillmann	B102	V	2 / 211

Abbildung 19: Ausgabe

5.3. Verarbeitung der Daten

Nachdem die Suchoptionen zur weiteren Auswertung übermittelt sind, wird die URL der Webseite zusammengesetzt.
Diese Zusammensetzung wird durch ein Array realisiert, welches zum Teil aus statischen Werten, wie zum Beispiel dem Pfad und die Bezeichner für die Übergabevariablen bestehet und zum Teil aus Werten, die zur Programmlaufzeit generiert werden, wie die Werte der Übergabevariablen. Das Array wird in einer Schleife ausgelesen. Dabei wird der String für die URL erzeugt. Mit dem PHP-Befehl *file(‚url')* wird dann die Webseite Aufgerufen.
Die Antwort darauf ist ein String. In diesem ist der Quelltext der Webseite enthalten, welcher normalerweise die Webseite darstellt.
Der String wird zeilenweise in ein Textdokument geschrieben.
In der Abbildung 20 ist ein Auszug dieses Textdokuments zu sehen, bei dem bereits die HTML-Tags mit dem PHP-Befehl *strip_tags(‚text')* entfernt wurden. Es enthält, neben einigen Elementen der Webseite, wie den Kalender, alle Veranstaltungen des Semesters für den Studiengang.

```
Dienstag, 16. September 2008

08:15-09:45
Hillmann: B102 Geostatistik GI 1.Sem VHaus 2 Raum
109
 
 

12:30-14:00
Kresse: B105 Grundlagen der
Geoinformatik GI 1.Sem VHaus 2 Raum 211
 
 
```

Abbildung 20: Auszug aus Antworttextfile

Die erstellte Funktion *parser_block()* sucht jetzt dieses File nach dem gesuchten Datum ab. Ist dieses gefunden, wird mit Hilfe von regulären Ausdrücken der nächste auftretende String gesucht, welcher das Format ([0-3][0-9])\. ([A-Z][a-zäöü]+) ([2][0][0-9][0-9]) (Bsp: 17. September 2008) hat beziehungsweise das Dokumentende erreicht ist.
Dieser Teil wird mit dem PHP-Befehl *substr()* aus dem File ausgeschnitten und zur weiteren Zerlegung an die Funktion *parser_veranstaltung()* weiter gegeben.
Diese Funktion trennt zunächst den String an den Semikola auf und prüft wiederum die Textpassage mit Hilfe der regulären Ausdrücke nach auftretenden Mustern. Die Ergebnisse werden in einem Array sortiert und zur Ausgabe weitergeleitet.

5.4. Datenorganisation

Um die Übersichtlichkeit des Quelltextes zu gewährleisten und um Erweiterungen möglichst einfach einbinden zu können, wurde der Programmcode auf 5 PHP-Files aufgeteilt (Abbildung 21).
Die Files *index.php*, *kurssuche.php* und *ausw.php* dienen dabei dem Programmablauf. Bei c*onfig.php* und var_lib.php handelt es sich um Programmteile zur Bereitstellung von globalen Funktionen und Variablen.

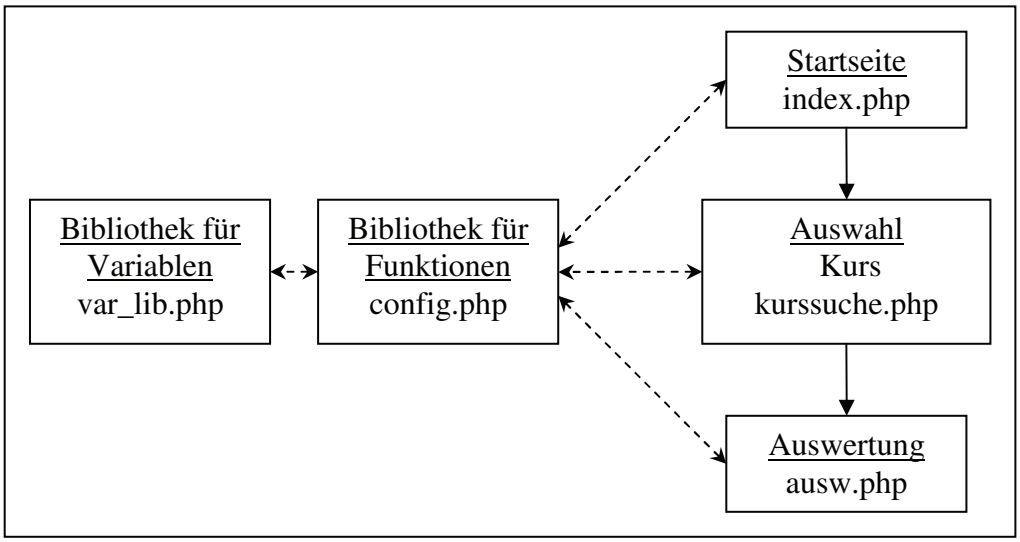


Abbildung 21: Datenorganisation

Index.php:
Das File dient als Startseite und soll Auswahlmöglichkeiten wie Suchfilter und Hilfe bereitstellen.

Kurssuche.php:
Der Code regelt die Informationsabfrage vom Nutzer, für die gezielte Ausgabe des Stundenplans.

Ausw.php:
Das File steuert die Auswertung der Anfrage. Die URL wird hier erzeugt, das Resultat empfangen, an die auswertenden Funktionen übergeben und ausgegeben.

Config.php:
Der Programmteil dient als Bibliothek für alle erstellten Funktionen. Jeder Programmteil greift auf diesen zu.

Var_lib.php:
Hierbei handelt es sich um eine Bibliothek für die Variablen. Hier befinden sich die Datenbankparameter, die regulären Ausdrücke und andere statische Variablen. Das File braucht nur in der config.php eingebunden zu werden, da alle anderen Fils auf config.php und damit auch auf var_lib.php zugreifen können.

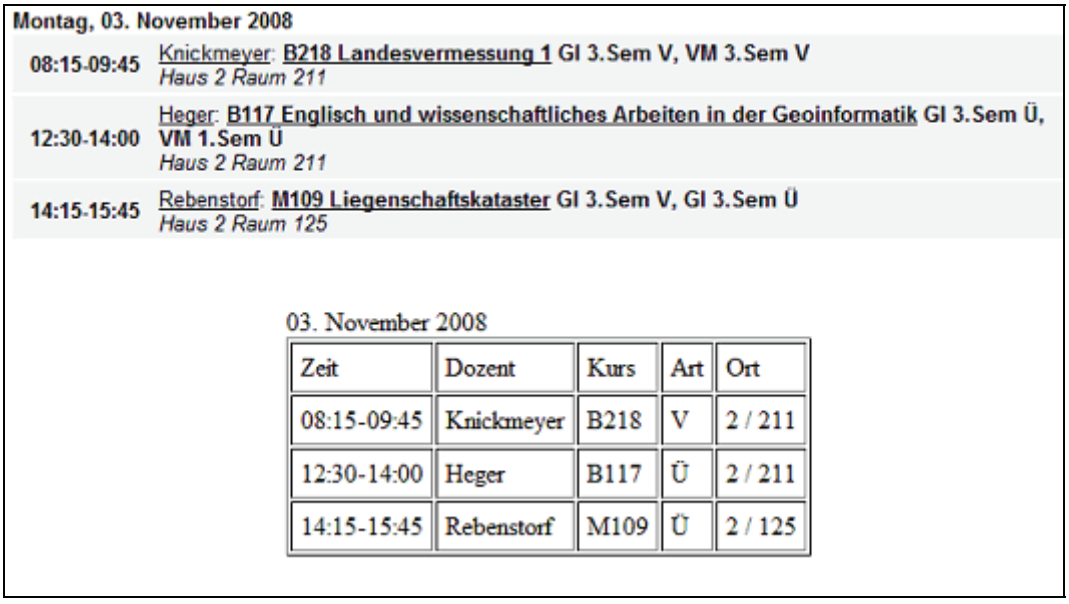
5.5. Datenbank

Die ursprüngliche Idee dieses Konzepts war es, eine lauffähige Lösung zu finden, die keiner eigenen Datenquelle bedarf. Im Laufe der Umsetzung hat sich jedoch gezeigt, dass auch hier eine Datengrundlage nötig ist um die Übersichtlichkeit und die Dynamik der WAP-Seite sicherzustellen.
Der Prototyp nutzt derzeit die in Abbildung 9 dargestellte Datenbank als Grundlage für die Zuweisung der Studiengänge zu den Fachbereichen. Diese erweist sich als ausreichend für den Entwurf.

5.6. Testphase

Um mögliche Fehler aufzudecken, wurde der Prototyp auf seine Funktionalität getestet. Dafür wurde mittels XAMPP eine Serverumgebung aufgebaut und eine MySql-Datenbank mit Beispieldaten angelegt.
Als Datensätze für die Tests erfolgte der Zugriff über das Internet auf die Daten des in der Testphase befindlichen Onlinestundenplans der Hochschule Neubrandenburg.
Die Funktionalität wurde mit dem Internetbrowser von Opera in der Version 9.62, mit Plug-In für XHTML, getestet.

Die Richtigkeit der Ausgabe fand im direkten Vergleich des WAP- und Webergebnis statt (Abbildung 22). Als Ausgangslage wurde Geoinformatik, 3. Semester, Gruppe A am 03.11.2008 genommen.
Der obere Teil der Abbildung zeigt dabei das Ergebnis der Webseite und der untere Teil die Ausgabe des Prototyps.



Montag, 03. November 2008

08:15-09:45	Knickmeyer: B218 Landesvermessung 1 GI 3.Sem V, VM 3.Sem V *Haus 2 Raum 211*
12:30-14:00	Heger: B117 Englisch und wissenschaftliches Arbeiten in der Geoinformatik GI 3.Sem Ü, VM 1.Sem Ü *Haus 2 Raum 211*
14:15-15:45	Rebenstorf: M109 Liegenschaftskataster GI 3.Sem V, GI 3.Sem Ü *Haus 2 Raum 125*

03. November 2008

Zeit	Dozent	Kurs	Art	Ort
08:15-09:45	Knickmeyer	B218	V	2 / 211
12:30-14:00	Heger	B117	Ü	2 / 211
14:15-15:45	Rebenstorf	M109	Ü	2 / 125

Abbildung 22: Gegenüberstellung der Ergebnisse

Allerdings stieß der Entwurf sehr schnell an seine Grenzen. Im Laufe der Tests traten Probleme auf, welche im Vorfeld nicht bedacht wurden. Zum Beispiel bestehen Unterschiede bei den Übergabeparametern der Webseite. So haben zwar die jeweiligen Semester der Bachelorstudiengänge die gleichen Semester ID aber die des Masterstudiengangs weicht davon ab.

Beispiel:
Die Bachelorstudiengänge Geoinformatik und Vermessungswesen haben für das 1. Semester die ID 8, der Masterstudiengang hingegen für das 1. Semester die ID 3.

Es handelt sich hierbei um einen Fehler bei der Planung und Umsetzung. Da die Übergabewerte der Semester im Programmcode fest eingebunden sind, erweist es sich als aufwendig diese Abweichungen im Quelltext abzufangen. Hier ist es Ratsam, das Problem bei der weiteren Entwicklung zu berücksichtigen und durch Erweiterungen der Datenbank zu umgehen.

Weitere kleine Fehler traten bei der Darstellung einiger Daten auf. So zum Beispiel bei außerplanmäßigen Veranstaltungen, welche nicht in das Suchmuster der regulären Ausdrücke passen .

6. Fazit

Das Ziel dieser Arbeit war es, ein Konzept zu erstellen welches die mobile Anbindung eines Onlinestundenplans regelt. Dabei sollte die Möglichkeit gegeben sein, mit minimalem Aufwand auch an andere Stundenplansysteme andocken zu können.

Im Zuge der Arbeit wurden zunächst die aktuellen Gegebenheiten im Bezug auf den Mobilfunk und den Anwender geklärt. Mittels Beispiele wurde deutlich gemacht, auf welchem Weg Webseiten mobil erreicht werden können.
Anhand des Beispiels WAP wurden 2 Lösungsansätze entwickelt, um den Onlinestundenplan über das mobile Internet zu erreichen.

Der erste Lösungsansatz beschreibt eine Schnittstelle, welche die Informationen der Web-Datenbank aufbereitet und in eine eigene Datenquelle überführt.
Dabei wurden Aufbau und Funktion beschrieben. Eine prototypische Umsetzung erfolgte dabei allerdings nicht.

Der zweite Lösungsansatzes ergab einen Prototyp zur Abfrage der Webinhalte. Dieser greift über die URL der Webseite auf die benötigten Daten zu, wandelt diese in das benötigte Format um und gibt sie auf dem Browser aus.
Der Prototyp ist lauffähig aber weist noch einige Fehler auf welche behoben werden müssen.
Eine mögliche Erweiterung für das Programm wäre eine Filteroption für die Dozentensuche. Somit wäre auch den Dozenten eine Möglichkeit der mobilen Stundenplanabfrage geboten.

Quellenverzeichnis

[1] Hochschul-Informations-System-GmbH, http://www.his.de/abt1/ab10
(Abgerufen am 03.05.2009)
[2] BITKOM, http://www.bitkom.org/de/presse/30739_57785.aspx
(Abgerufen am 03.05.2009)
[3] Wipo.int, http://www.wipo.int/pctdb/en/wo.jsp?IA=DE2002004176&DISPLAY=DOCS
(Abgerufen am 03.05.2009)
[4] World Wide Web Consortium, http://www.w3.org/TR/xhtml-basic/
(Abgerufen am 03.05.2009)
[5] Sun Microsystems, http://java.sun.com/javame/index.jsp
(Abgerufen am 03.05.2009)
[6] DBKonzern, http://www.bahn.de
(Abgerufen am 03.05.2009)
[7] Google Inc., http://www.google.de
(Abgerufen am 03.05.2009)
[8] Ebay, http://www.ebay.de
(Abgerufen am 03.05.2009)

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Anhang A

Index.php

```
<?php

  /* Startseite
  Auswahlmenü für verschiedene Suchoptionen  */

  /* Einbindung config.php für Zufriff auf Funktionen */

  require_once('config.php');

  kopfzeile();

  echo "Willkommen im mobilen Stundenplan der Hochschule Neubrandenburg <br>";
  echo "Bitte wählen Sie die gewünschte Suchoption aus: <br><br><br>";

  echo "<table>";
  echo "<tr><td>";

  /* Button Kurssuche mit weiterer Funktion hinterlegt */

  echo "<form action='kurssuche.php' method='GET'>";
  echo "<input type='SUBMIT' value='Kurssuche'>";
  echo "</form></td></tr>";
  echo "<tr><td>";

  /* Dozentensuche und Hilfe noch ohne Funktion, dienen als Platzhalter für

erweiterungen*/

  echo "<form action='dozentsuche.php' method='GET'>";
  echo "<input type='SUBMIT' value='(Dozentensuche)'>";
  echo "</form></td></tr>";
  echo "<tr><td>";
  echo "<form action='hilfe.php' method='GET'>";
  echo "<input type='SUBMIT' value='(Hilfe)'>";
  echo "</form></td></tr>";
  echo "</table>";

  fusszeile();

?>
```

kurssuche.php

```
<?php

    /* Kurssuche
      Formular, welches die Auswahl des Fachbereichs, Studiengangs, Semester, Gruppe
      und Datum fordert*/

    require_once('config.php');

    kopfzeile();

    /* Formular für Auswahl des Fachbereichs
          ruft sich selbst mit der Get-Methode auf
          und erkennt, anhand "Var1", ob Fachbereich gewählt wurde*/

  $var1 = $_GET['Var1'];

  if ($var1 != 1){

  echo "<table border='0'>";
  echo "    <form action= 'kurssuche.php' method='GET'>";
  echo "      <tr><td>";
  echo "        <select size='1' name='S1'>";
  echo "          <option value = '1'> LGGB </option>";
  echo "          <option value = '2'> AWLW </option>";
  echo "          <option value = '3'> SW</option>";
  echo "          <option value = '4'> </option>";
  echo "          <option value = '5'> </option>";
  echo "        </select>";
  echo "      </td><td>";
  echo "      <input type='hidden' value='1' name='Var1'>";
  echo "      <input type='submit' value='OK'>";
  echo "      </td></tr>";
  echo "    </form>";
  echo "</table>";

  }else{

  /* Wenn Fachbereich gewählt, Datenbank auslesen für Studiengang */

        $FB = $_GET['S1'];

        $sql = "SELECT * FROM Studiengang WHERE FBID=$FB";
        $ergebnis = Datenbank ($sql);

        echo "<table border='1' cellpadding='5'>";

        $Rz = mysql_num_rows($ergebnis);
        $Cz = mysql_num_fields($ergebnis);
```

```
$j=0;

/* Formular erzeugen für Studiengangauswahl
            Auswahl wird mit GET an ausw.php gesendet */

echo "<form action= ausw.php method='GET'>";

echo "<tr><td>";
echo "Auswahl Studiengang: </td><td>";

echo "<select size='1' NAME= 'Studiengang'>";
echo "<option value= 'KO' > </option>";

for ($i=0;$i<$Rz;$i++){
for ($k=0;$k<$Cz;$k++){
   $feld[$i][$k] =(mysql_result($ergebnis,$i,$k) . "");
}

$x[$j] = $feld[$i][0];
$y[$j] = $feld[$i][1];

echo "<option value= '$x[$j]' > " . $y[$j] . " </option>";

$j++;
}

echo "</select>";
echo "</td></tr>";
echo "<tr><td>";

/* Auswahlbox für Fachsemester */

echo "Auswahl Semester: </td><td>";

echo "<select size='1' NAME= 'Semester'>";
echo "<option value= " > </option>";
echo "<option value= '8' >  1. Semester  </option>";
echo "<option value= 'a' >  2. Semester  </option>";
echo "<option value= '9' >  3. Semester  </option>";
echo "<option value= 'a' >  4. Semester  </option>";
echo "<option value= '2' >  5. Semester  </option>";
echo "<option value= 'a' >  6. Semester  </option>";
echo "<option value= 'a' >  7. Semester  </option>";
echo "<option value= 'a' >  8. Semester  </option>";
echo "</select>";

echo "</td></tr>";
echo "<tr><td>";

/* Auswahlbox für Gruppe */
```

```
        echo "Auswahl Gruppe (optional):  </td><td>";
        echo "<select size='1' NAME= 'Gruppe'>";
        echo "<option value= " > </option>";
        echo "<option value= '9' >  Gruppe A  </option>";
        echo "<option value= '10' >  Gruppe B  </option>";
        echo "<option value= '11' >  Gruppe C  </option>";
        echo "<option value= '12' >  Gruppe D  </option>";
        echo "<option value= '13' >  Gruppe E  </option>";
        echo "<option value= '14' >  Gruppe F  </option>";
        echo "</select>";
        echo "</td></tr>";

        /* Textfelder für Eingabe des Datum*/

        echo "<tr><td>Datum:</td><td>";
        echo "<input type='text' NAME='Dat1' size='2' maxlength='2' value='16'>. ";
        echo "<input type='text' NAME='Dat2' size='2' maxlength='2' value='09'>. ";
        echo "<input type='text' NAME='Dat3' size='4' maxlength='4' value='2008'> ";
        echo "</td></tr><tr>";
        echo "<td></td><td>";
        echo "<input type='submit' Value='OK'>";
        echo "</td></tr>";
        echo "</form>";
        echo "</table>";
        }

   fusszeile();

?>
```

ausw.php

```
<?php

require_once("config.php");
kopfzeile();
echo $datum;

/* Erzeugung der WWW-URL auf Grundlage der var_lib.php */

$link = $lnk[0];

for ($i = 1; $i<count($lnk);$i++){
        if ($val[$i] != "") $link = $link . $lnk[$i] . $val[$i];
}

/* Aufruf der WWW-URL */

$feld = file($link);
$datei = fopen ("test1.txt","w+");

/* Auslesen der WWW-URL zeilenweise und schreiben in Textfile, HTML-Tags entfernen
   Es ist besser die Daten direkt zu übergeben, als erst in File zwischenzuspeichern
   ist zu Testzwecken aber so einfacher */

foreach ($feld as $line_num => $line) {

        fwrite($datei, strip_tags(($line)));
}

fclose($datei);

/* Übergabe des Suchdatums an Auswertemechanismus in config.php*/
$resultat = parser_veranstaltung(parser_block($datum));

//  unlink("test1.txt");
/* Ausgabe der gewünschten Werte */
echo "<table border='1' Cellpadding='5'>";
echo "<tr><td>Zeit</td><td>Dozent</td><td>Kurs</td><td>Art</td><td>Ort</td>";
for ($i=0;$i<count($resultat);$i++){
if (ereg($regexp->time,$resultat[$i]) == true ){
                echo "</tr><tr><td> $resultat[$i] </td>";
        }else{
                echo "<td> $resultat[$i] </td>";
        }
}
echo "</tr></table><br>";
// echo $link;   // für Test
fusszeile();
?>
```

config.php

```
<?php

/* Stellt eine Reihe von Funktionen bereit */

include('var_lib.php');
function kopfzeile(){
/* Stellt den Header für die XHTML Seiten bereit */

echo "<!doctype html public '-//W3C//DTD XHTML Basic 1.0 //EN'";
echo "'http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd'>";
echo"<html>";
echo"<head>";
echo"<title>'Mobiler Stundenplan der HS-NB'</title>";
echo"</head>";
echo "<body>";
}

function fusszeile(){

        /* Schließende Tags für das Fileende
           optional können hier noch Buttons für Navigation eingebunden werden
           Bsp: "Zurück", "Startseite" oder "Hilfe" */

        echo "</body>";
        echo "</html>";
}

function Datenbank($sql){

   /* Fünktion, die den Datenbankaufruf und die Abfrage regelt
  erwarteter Wert ($SQL) ist das SQL-Statement
  Rückgabewert ($value) ist der Ergebnis-Handle */

  include('var_lib.php');

  $db = mysql_connect($dbc->dbhost, $dbc->dbuser, $dbc->dbpass)
          or die ("keine Verbindung möglich");
  mysql_select_db ($dbc->dbname , $db);
  $value = mysql_query ($sql, $db) or die (mysql_error());
  return $value;
}

function parser_block($datum){

  include('var_lib.php');

  /* Zum umwandeln des Quelltextes der angesprochenen Webseite für die Ausgabe.
  das Textfile wird in einen Block für das gesuchte datum zerlegt
  übergebener Wert ($datum) ist das Suchdatum
```

```
  Rückgabewert ($ergebniss) ist ein String in dem alle Veranstaltungen für das
  gesuchte Datum im gesuchten Studiengang enthalten sind */

        // erzeugtes file öffnen und auswerten

        $auswertung = fopen ("test1.txt","r+");
        $text=fread($auswertung, 20000000);
        fclose ($auswertung);
        $p1 = strpos($text, $datum);

        /** weitersuchen nach dem nächsten auftreten eines Datums **/

        preg_match($regexp->date, $text, $treffer1, PREG_OFFSET_CAPTURE, $p1+10);

        $p2 = $treffer1[0][1];
      // echo " nächster tag: " . $treffer1[0][0]; //füt test

        if ($p2 == 0){
        $ergebnis= substr($text, $p1);
        }else{

        $p3= $p2 - $p1;
        $ergebnis= substr($text, $p1, $p3);
        }

        $len = strlen($datum);
        $ergebnis = substr($ergebnis, $len) ;

        return $ergebnis;
}

function parser_veranstaltung($block){
  /* Funktion zum Zerlegen des übegebenen Strings ($block)
     in die benötigten Bestandteile und Rückgabe als Array ($block_erg)*/

include('var_lib.php');

  /* Zerlegen des Strings bei auftretenden ";" und speichern in Array */
  $block = explode(";" , $block );
  $j = 0;

  /* Test der einzelnen Felder nach Auftreten von Mustern regulärer Ausdrücke
   Wenn erfolgreich, dann ablegen in Ergebnis-Array  */

  for ($i=0;$i<count($block);$i++) {

  $test  = ereg($regexp->time , $block[$i]);
  $test1 = ereg($regexp->dozent , $block[$i]);
  $test2 = ereg($regexp->modul , $block[$i]);
  $test3 = ereg($regexp->art , $block[$i]);
  $test4 = ereg($regexp->haus , $block[$i]);
```

```
  if ($test == true) {
  /* Sucht nach der Zeit */
  preg_match($regexp->time, trim($block[$i]), $treffer, PREG_OFFSET_CAPTURE);
  $block_erg[$j] = $treffer[0][0];
  $j++;
  }
  if ($test1 == true) {
  /* Sucht nach Dozent */
  preg_match($regexp->dozent, trim($block[$i]), $treffer, PREG_OFFSET_CAPTURE);

  if ($treffer[0][0] == "Pause:"){
  preg_match($regexp->dozent, trim($block[$i]), $treffer1, PREG_OFFSET_CAPTURE,
  $treffer[0][1]+5);
  $block_erg[$j] = strtr($treffer1[0][0],":"," ");
  $j++;

  }else{

  $block_erg[$j] = strtr($treffer[0][0],":"," ");
  $j++;
  }
  }

  if ($test2 == true) {

  /* Sucht nach Modulnummer */
  preg_match($regexp->modul, trim($block[$i]), $treffer, PREG_OFFSET_CAPTURE);
  $block_erg[$j] = $treffer[0][0];
  $j++;
  }

  if ($test3 == true) {
  /* Sucht nach Veranstaltungsart */
  preg_match($regexp->art, trim($block[$i]), $treffer, PREG_OFFSET_CAPTURE);
  $block_erg[$j] = Strtr($treffer[0][0],"H"," ");
  $j++;
  }

  if ($test4 == true) {
  /* Sucht nach Haus & Raum */
  preg_match($regexp->haus, trim($block[$i]), $treffer, PREG_OFFSET_CAPTURE);
  $block_erg[$j] =Trim(Strtr(Strtr($treffer[0][0],"Hausm","       "),"R","/"));
  $j++;
  }
  }

  /* Rückgabe zur Auswertung */
  return $block_erg;
}
?>
```

var_lib.php

```
<?
/* - dient der Definition von genutzen statischen Variablen
   - erleichtert die Änderungen
   - zentrale Variablenverwaltung */

    /* Definition der Regulären Ausdrücke zur Zerlegung der empfangenen
    HTML-Passagen */

    $regexp->time = '([0-9]{2}[:][0-9]{2}[-][0-9]{2}[:][0-9]{2})';
    $regexp->dozent = '([A-ZÄÖÜ][a-zäöü]+[:])';
    $regexp->modul = '([A-Z][0-9]{3})';
    $regexp->art = '(([A-ZÄÖÜ]|[A][n][o][E][t]|[B][r][ü][a-z]*)([H]))';
    $regexp->haus = '(([H][a-z]+) ([0-9]+) ([R][a-z]+) ([0-9]+))';
    $regexp->date = '(([0-3][0-9])\. ([A-Z][a-zäöü]+) ([2][0][0-9][0-9]))';

    /* Festlegung der Parameter für integrierte Datenbank */

    $dbc->dbname = "Stundenplan_mob“;
    $dbc->dbhost = "localhost";
    $dbc->dbuser = "SPmob_Admin";
    $dbc->dbpass = "SPmob_Admin";

        /* Abfrage des Suchdatum und Formatierung */

    $monat = $_GET['Dat2'];

    switch($monat) {
        case 1:
          $monat = "Januar";
          break;
        case 2:
          $monat = "Februar";
          break;
        case 3:
          $monat = "März";
          break;
        case 4:
          $monat = "April";
          break;
        case 5:
          $monat = "Mai";
          break;
        case 6:
          $monat = "Juni";
          break;
        case 7:
          $monat = "Juli";
          break;
        case 8:
```

```
  $monat = "August";
  break;
 case 9:
  $monat = "September";
  break;
 case 10:
  $monat = "Oktober";
  break;
 case 11:
  $monat = "November";
  break;
 case 12:
  $monat = "Dezember";
  break;
 default:
  echo "";
  }

  $datum = $_GET['Dat1'] . '. ' . $monat . ' ' . $_GET['Dat3'];

  /* Parameter für URL zum Webinhalt */

  $lnk[0] = "http://portal.hs-nb.de/~stundenplan/index.php";
  $lnk[1] = "?org=";
  $lnk[2] = "&jhg=";
  $lnk[3] = "&sem=";
  $lnk[4] = "&gru=";
  $lnk[5] = "&vordefsuche=";

  $val[0] = "";
  $val[1] = $_GET['Studiengang'];
  $val[2] = $_GET['Semester'];
  $val[3] = "2008_0";
  $val[4] = $_GET['Gruppe'];
  $val[5] = "jahrgang_gruppen_id";

?>
```

db_setup.php

```
<?php

  /* wurde Angelegt, um eine Datenbank als Grundlage mit Dummiedaten zu definieren */

 require_once ('var_lib.php');

// Datenbank mit Standardnutzer "root" ansprechen

$db = mysql_connect($dbc->dbhost, 'root') or die ('Keine Verbindung zur Datenbank!');

// Datenbank anlegen

$sql1 = 'CREATE DATABASE '.$dbc->dbname." ;
$value1 = mysql_query ($sql1, $db) or die (mysql_error());
echo $value1 . "<br>";

// Datenbanknutzer anlegen

$sql2 = "CREATE USER '$dbc->dbuser'@'localhost' IDENTIFIED BY '$dbc->dbpass';";
$value2 = mysql_query ($sql2, $db) or die (mysql_error());
echo $value2 . "<br>";

// Rechte für Nutzer zuteilen

$sql3 = "GRANT ALL PRIVILEGES ON `$dbc->dbname` . * TO '$dbc->dbuser'@'localhost'
WITH GRANT OPTION ; ";
$value3 = mysql_query ($sql3, $db) or die (mysql_error());
echo $value3 . "<br>";

// Angelegte Datenbank mit neuem Nutzer öffnen

$db = mysql_connect($dbc->dbhost, $dbc->dbuser, $dbc->dbpass ) or die ('Keine

Verbindung zur Datenbank!');
mysql_select_db ($dbc->dbname , $db);

// Tabellen anlegen

$sql4 = "CREATE TABLE Fachbereich (";
$sql4.= "FBID INT( 50 ) NOT NULL AUTO_INCREMENT PRIMARY KEY , ";
$sql4.= "Name VARCHAR( 500 ) NOT NULL , ";
$sql4.= "Kurz VARCHAR( 50 ) NOT NULL ) ";
$sql4.= "ENGINE = MYISAM ";
$value4 = mysql_query ($sql4, $db) or die (mysql_error());
echo $value4 . "<br>";

$sql5 = "CREATE TABLE Studiengang (";
$sql5."SGID INT( 50 ) NOT NULL AUTO_INCREMENT PRIMARY KEY , ";
$sql5."Name VARCHAR( 500 ) NOT NULL , ";
```

```
$sql5.="Kurz VARCHAR( 50 ) NOT NULL , ";
$sql5.="FBID INT( 50 ) NOT NULL ) ";
$sql5.="ENGINE = MYISAM ";

$value5 = mysql_query ($sql5, $db) or die (mysql_error());

echo $value5 . "<br>";

$sql6 = ("INSERT INTO `$dbc->dbname`.`Fachbereich` (`FBID` ,`Name` , `Kurz`)
VALUES ("

, 'Landschaftsarchitektur und Geoinformatik' , 'LGGB' );");
$value6 = mysql_query ($sql6, $db) or die (mysql_error());
$sql7 = ("INSERT INTO `$dbc->dbname`.`Fachbereich` (`FBID` ,`Name` , `Kurz`)
VALUES ("

, 'Agrarwirtschaft und Lebensmittelwisschschaften' , 'AWLW' );");
$value7 = mysql_query ($sql7, $db) or die (mysql_error());
$sql8 = ("INSERT INTO `$dbc->dbname`.`Fachbereich` (`FBID` ,`Name` , `Kurz`)
VALUES ("

, 'Soziale Arbeit, Bildung und Erziehung' , 'SBE' );");
$value8 = mysql_query ($sql8, $db) or die (mysql_error());
$sql9 = ("INSERT INTO `$dbc->dbname`.`Studiengang` (`SGID` ,`Name` , `Kurz` , `FBID`)

VALUES (" , 'Geoinformatik' , 'GI' , '1' );");
$value9 = mysql_query ($sql9, $db) or die (mysql_error());
$sql10 = ("INSERT INTO `$dbc->dbname`.`Studiengang` (`SGID` ,`Name` , `Kurz` , `FBID`)

VALUES (" , 'Vermessungswesen' , 'VM' , '1' );");
$value10 = mysql_query ($sql10, $db) or die (mysql_error());
$sql11 = ("INSERT INTO `$dbc->dbname`.`Studiengang` (`SGID` ,`Name` , `Kurz` , `FBID`)

VALUES (" , 'Geodäsie und Geoinformatik' , 'GG' , '1' );");
$value11 = mysql_query ($sql11, $db) or die (mysql_error());
$sql12 = ("INSERT INTO `$dbc->dbname`.`Fachbereich` (`FBID` ,`Name` , `Kurz`)
VALUES

(" , 'Fachbereich Gesundheit, Pflege, Management' , 'GP' );");
$value12 = mysql_query ($sql12, $db) or die (mysql_error());

?>
```

Anlage B

ReadMe_SPmob.txt

Informationen zur Einbindung und Nutzung des Prototyps zur Stundenplanabfrage

1. Voraussetzungen
2. Einbinden des Quellcode
3. Datenbank erstmalig anlegen
4. Startseite aufrufen

****
1. Voraussetzung

- Arbeitsplatz mit Internetanbindung
- Server mit MySql-Datenbank (XAMPP)
- Internetbrowser
- (optional: Emulator von XHTML-mobile)

****
2. Einbinden des Quellcode

- Das Verzeichnis ... in den Webserver einfügen
  (bei XAMPP -> /xampp/htdocs)

****
3. Datenbank erstmalig anlegen

Die mitgelieferte Datenbank enthält Testdaten zur Demonstration der Funktionalität.
Es wird automatisch angelegt:

- Datenbank : "Stundenplan_mob"
- Nutzer : "SPmob_Admin"
- Nutzerpasswort: "SPmob_Admin"
- Nutzerrechte : Alle Rechte für Stundenplan_mob
- Tabelle 1 : "Fachbereich" mit 3 Spalten und Testeinträgen"
- Tabelle 2 : "Studiengang" mit 4 Spalten und Testeinträgen"

Um die Datenbank anzulegen muss das File db_setupt.php im Browser aufgerufen werden.
Die Datenbank wird automatisch erstellt. Erscheint 5-mal die Ziffer "1" wurde die
Datenbank erfolgreich angelegt und kann verwendet werden.

Link für Aufruf: http://localhost/SPmob/db_setup.php

****
4. Startseite aufrufen

- Aufruf erfolgt dann weiterhin über den Browser

Link für Aufruf: http://localhost/SPmob/index.php
****