mobileHdM

Wintersemester 2010 | 2011

>> Entwicklung

Orlando Schäfer (os019)
Andreas Petsch (ap042)
Simon Lipke (sl064)

>> Projekt Management

Boris Steiner (bs065)

>> Versionierung

Name	Version	Anmerkung	Datum
Simon Lipke	0.1	Ersterstellung	03.01.11
Simon Lipke	0.2	Webbackend	10.01.11
Simon Lipke	0.3	Plugins & Webbackend	12.01.11
Simon Lipke	0.4	Pflichtenheft v0.8 & Bearbeitung mehrerer Artikel	12.02.11
Orlando Schäfer	0.5	Bearbeitung mehrerer Artikel	13.02.11
Andreas Petsch	0.6	Bearbeitung mehrerer Artikel	13.02.11
Orlando Schäfer	0.7	Layout Storyboard Mockups	17.02.11
Orlando Schäfer	0.8	Anhang Bearbeitung mehrerer Artikel	18.02.11
Andreas Petsch	0.9	Bearbeitung mehrerer Artikel	20.02.11
Orlando Schäfer	1.0	Detailbearbeitungen Abschluss	21.02.11

>> Einleitung

Erstsemester an der Hochschule der Medien haben es nicht leicht.

Insbesondere in Hinsicht darauf, wo und wie sie schnell an die wichtigsten Informationen gelangen. An der Hochschule existieren mehrere Informationskanäle, die allesamt wichtige Informationen zum Studienverlauf, zur Studienorganisation und sonstigen Tätigkeiten der HdM liefern.

Jeder Professor entscheidet sich für einen solchen Kanal, um seine Informationen zu veröffentlichen und bis heute gibt es keine einheitliche Methode, alle wichtigen Informationen von allen Professoren auf einen Blick zu erhalten.

„**mobileHdM**“ ist eine iPhone Anwendung, die es HdM-Studenten einfach ermöglicht, Zugriff auf alle gängigen Kommunikationskanäle und Informationsquellen der HdM zu erlangen, mit nur einem einzigen Login.
Insbesondere Erstsemestern wird somit der Einstieg in die HdM wesentlich erleichtert, denn es muss nicht für jede Quelle ein zusätzliches Programm installiert und konfiguriert werden oder über den Browser angesteuert werden.

>> Motivation

Wir haben also gelernt, dass die Informationen der HdM weit gestreut sind. Wie aber kamen wir konkret auf die Idee für „**mobileHdM**“?

Dazu muss man sich in den Alltag versetzen.

Was sind die Fragen, die sich ein HdM-Student (und sicher auch Studenten anderer Hochschulen) täglich stellt?

Da wäre zum Beispiel „**Hat ein Professor in der Newsgroup was wichtiges gepostet?**“. Sicherlich bekannt sind hier zu Ende eines Semesters die Dauer-Einlog-Orgien um in regelmäßigen Abständen die offizielle Newsgroup abzurufen. Eventuell hat einer der Professoren ja wieder etwas wichtiges zum Studienablauf oder sogar Noten online gestellt.

Eine weitere klassische Frage ist: „**Was gibt es heute in der Mensa zu essen?**“. Zur Mittagszeit ruft der Hunger und dabei ist man schon aus dem Vorlesungssaal hinausgelaufen, sodass man keine Lust hat, wieder umständlich seinen Laptop aus der Tasche zu packen um auf die Mensa-Seite zu surfen.

Eine Situation, die meistens zu Anfang eines Semesters auftritt: Man kommt morgens an der HdM an und hat ganz vergessen, **in welchem Raum man die erste Vorlesung** hat. Wirres Kommilitonen-Anrufen und SMS mit dem Satz „Wo haben wir jetzt?“ verschicken ist hier der Fall.

Oder wer kennt das nicht? Man sitzt an einer Email, hat diese fast zu Ende geschrieben, möchte den Kommilitonen oben in die „An:“-Zeile eintragen und fragt sich: „**Wie war denn nochmal sein Kürzel?**“

Damit eben genau diese Fragen, die einen im ständigen Studenten-Alltag begleiten, einem nicht mehr so nervig erscheinen, haben wir uns für eine mobile und schnelle Variante der Problemlösung entschieden.

Wir haben uns für „**mobileHdM**“ entschieden.

>> Projektdefinition

> Projektziel

Das Ziel unseres Projektes ist, allen HdM Studenten eine einfache Möglichkeit zu bieten, auf die gängigen Informationskanäle der HdM mit einem SSO (Single-Sign-On) zu ermöglichen. Außerdem soll mithilfe unseres Webbackends die Möglichkeit gegeben sein, Informationen zusätzlich einfach auf andere mobile Plattformen, wie z.B. Android, zu schicken.

> Zielgruppe

Die Anwendung wird in erster Linie von HdM-Studenten benutzt werden, um mobilen Zugriff auf die gewünschten Daten zu erhalten. Beschränkt wird die Benutzung auf Besitzer eines iPhone / iPod-Touch mit Zugriff auf mobiles Internet oder WLAN.

> Anforderungen

Betriebsbedingungen

Für die Nutzung der Anwendung wird zunächst ein iPhone / iPod-Touch mit einer bestehenden Internetverbindung benötigt (GPRS/EDGE/UMTS oder WLAN).

Nichtfunktionale Anforderungen

Usability

Eines unserer Hauptziele ist es, mit einer einfachen Bedienung und wenig Internettraffic, mobil und schnell die gesuchten Informationen zu erlangen.
Um die wichtigsten Funktionen zu erreichen (Newsgroups, Stundenplan, Kürzel-Kai oder Speiseplan) sind weniger als 4 Klicks notwendig.

Performance

Durch die Optimierung und der lokalen Speicherung der Daten (Caching, Zwischenspeichern), wird der Internettraffic auf ein Minimum reduziert und somit die Performance deutlich erhöht. Somit ist bei einer schnellen WLAN Verbindung (und nicht Zwischengespeicherten Daten) eine Reaktionszeit von weniger als 2 Sekunden möglich. Bei bereits Zwischengespeicherten Daten ist eine Reaktionszeit von weniger als 1 Sekunde zu erwarten, da lediglich die GUI aufgebaut werden muss. Bei einer langsameren EDGE / GPRS Verbindung, muss mit einer Reaktionszeit von ~10 Sekunden gerechnet werden, da die Daten zunächst aus dem Internet geladen werden müssen.

Apple-Guidelines

Die von Apple vorgegebenen Richtlinien zur Gestaltung des Interfaces (Navigierbarkeit) oder der eigentlichen Programmierung (z.B. Speicherverwaltung) werden eingehalten, mit dem Ziel den Apple-Review Prozess zu bestehen.

> Projektumfang

Im Folgenden werden die Muss-, Soll- und Kann-Kriterien des Projektes beschrieben. Muss- und Soll-Kriterien sind diejenigen, die implementiert werden müssen, um das Projekt erfolgreich abzuschließen, hierbei haben Soll-Kriterien lediglich eine geringere Priorität. Kann-Kriterien hingegen beeinflussen nicht den erfolgreichen Abschluss des Projektes, sondern sind nur wünschenswert.

Muss-Kriterien

Im Folgenden werden die Muss-Kriterien im Detail aufgelistet.

Webbackend

Mittels PHP werden die Daten der einzelnen Bezugspunkte auf einem zentralen Webserver gesammelt. Dies soll einem Plugin-System ähneln, sodass in der Zukunft weitere Informationsquellen, die für die Realisierung von hilfreichen Anwendungen nützlich sind, einfach hinzugefügt werden können.
Um die Daten im weiteren Schritt zur iPhone Applikation (und für spätere Projekte - auch auf jede andere Plattform) zu transferieren, wird die Technik XML verwendet. Backend Features können nach belieben an- bzw. ausgeschalten werden. Sollte ein Plugin deaktiviert werden oder nicht verfügbar sein, wird beim Client anstatt des Contents eine entsprechend verständliche Fehlermeldung angezeigt.

Newsgroups

Der Newsgroupzugriff soll in den Muss-Kriterien erst einmal nur mit Lesezugriff realisiert werden. Im ersten Schritt müssen die gewünschten Newsgroups abonniert werden, diese können dann in der Gesamtübersicht einzeln angeklickt werden und die Themen in der einzelnen Gruppe angezeigt werden.

Stundenplan

Der Zugriff auf den Stundenplan wird zunächst ebenfalls nur lesend erfolgen. Die einzelnen Veranstaltungen werden mit ihrer Raumnummer tabellarisch nach Wochentag (x-Richtung) und Uhrzeit (y-Richtung) geordnet. Durch Auswahl einer bestimmten Veranstaltung aus dem Wochenplan werden detailliertere Informationen angezeigt: Uhrzeit, Dozent und Raumnummer. Der Stundenplan wird als einfacher Webzugriff in die iPhone Anwendung implementiert, da dieser bereits in einer Version für mobile Endgeräte im SS10 als Projekt entwickelt wurde.

Speiseplan

Der Speiseplan wird einfach und übersichtlich pro Tag nach Kalenderwoche dargestellt. Durch Auswahl eines bestimmten Datums, kann der Benutzer in einer Tabelle die jeweiligen Speisen nebst Preisen für diesen Tag finden. Die Speisen sind außerdem in Sektionen für die S-Bar und die Mensa unterteilt

Soll-Kriterien

Im Folgenden werden die Soll-Kriterien im Detail besprochen.

Kürzel-Kai

Der Kürzel-Kai ermöglicht es, den richtigen Namen hinter einem HdM-Kürzel zu erfragen und umgekehrt.

Kann-Kriterien

Die weiteren, hier aufgelisteten Anbindungen fallen unter die "Nice-to-have"-Kategorie.

Messi Anbindung

Messi ist ein HdM-internes Instant Messaging System, welches der Kommunikation zwischen den HdM-Studenten dient. Automatisch sind alle Kommilitonen desselben Studiengangs und Semesters in einer Freundes-Liste verfügbar.
In mobileHdM sollen diese über eine tabellarische Sicht angezeigt werden. Über die Auswahl eines bestimmten Freundes in der Liste öffnet sich eine Chat-Sicht und es können Nachrichten abgesetzt werden. Der Dialog ist oberhalb der Eingabe-Tastatur ersichtlich.

E-Mail
Über die E-Mail-Accounts erhalten die Studenten wichtige Informationen zu ihrem Studium wie Vortragsankündigungen, Raumänderungen, Termine, Fristen und Erinnerungen. Es ist sehr von Vorteil diese Informationen zu jeder Zeit abrufen zu können um immer auf dem neuesten Stand zu bleiben.

Professoren-Infos
Über die Teilanwendung "Professoren-Infos" können detaillierte Daten zu Professoren abgerufen werden, wie z.B. Sprechstunden, Raumnummer oder E-Mail Adresse.

Mobile HoRst
In einer lite-Version des Raumsuchprogrammes der HdM, können Studenten nach beliebigen Räumen der HdM suchen. In einer 2D-Ansicht wird die Lage des gewünschten Raumes angezeigt. Durch dieses Feature müssen Studenten nicht nach einem HoRst Terminal suchen, sondern können direkt von ihrem iPhone die Lage ihres Raumes herausfinden. Optional wäre auch eine Anbindung an den Vorlesungsplan denkbar. Durch einen Klick auf den Vorlesungsraum, wird dieser im HoRst gesucht und angezeigt.

Anbindung an SB-Funktionen
Die meistgenutzte Funktion in den SB-Funktionen für Studenten ist die Notenliste. Durch den Zugriff auf die SB-Funktionen können die Studenten mobil und mit nur wenigen Klicks ihre aktuelle Notenliste einsehen, und bekommen direkt bei Veröffentlichung der Noten eine Push-Nachricht zugeschickt. Die Prüfungsfächer werden in Semester aufgeteilt, übersichtlich aufgelistet und die erreichte Note mit angezeigt.

HdM Facebook / Twitter
In einer einfachen Ansicht können die Informationen der Social-Network Accounts der HdM abgerufen werden. Dazu gehören z.B. die Stundenplanänderungen über den Twitter Account oder die Facebook-News.

>> Konzeption

> Machbarkeitsprüfung und Evaluierung

Was vor Projektbeginn im Vordergrund stand, war eine Überprüfung, ob die von uns gesetzten Ziele in dem Projektzeitraum realisierbar sind und wenn ja, auf welche Weise. Automatisch ergaben sich hieraus auch jene Komponenten, die sich für die Zeitspanne als zu komplex erwiesen und deswegen aus der Planung herausgenommen werden mussten.

Da sich die Muss-Kriterien auf rein lesenden Zugriff beschränken sind diese von uns als überaus machbar eingestuft worden. Mit hinzu gerechnet wurde die Einarbeitungszeit in die Programmiersprache Objective-C und den Entwicklungstools (XCode + Interface Builder).

Außerdem musste der mobile Stundeplan von uns nicht von Grund auf neu programmiert werden, da sich im Vorsemester ein Projektteam bereits mit der Realisierung einer geeigneten mobilen Version des HdM Stundeplans bemüht hat. Somit ergab sich auch hier eine entscheidende Möglichkeit Zeit einzusparen, die für komplexere Gebiete nötig waren, wie zum Beispiel der Realisierung der Newsgroups.

Als besonders wichtig erachtet wurde die schnelle Implementierung des Webbackends, sodass bei der Entwicklung für das iPhone sehr früh mit Realdaten getestet werden konnte.

Da wir uns streng an die Apple Guidelines hielten, sind wir uns auch bei dem Design schnell einig gewesen und konnten anhand anderer ähnlich aufgebauten Apps (TwitterApp, Mail) unser Design schnell festlegen, sodass hier auch wenig Zeit benötigt wurde.

Ein von uns angedachtes Feature musste in der frühen Entwicklungsphase bereits aus unseren Plänen herausgenommen werden. Die Funktion, Noten der SB-Funktion der HdM einsehen zu können und bei Erhalt einer neuen Note diese mittels Push-Notification dem User zu zeigen, ist aufgrund politischer Gründe und fehlenden Rechten unsererseits hochschulseitig abgelehnt worden.

Testing und Evaluation geschah parallel zur Entwicklung. Den Entwicklern stehen Testgeräte zur Verfügung (iPhone 3G, 3GS und 4 ebenso wie iPod Touch 2G).
Zwischenzeitliche Deployments auf privaten Geräten von Mitstudenten und Betreuer, damit jene die App auch im Alltag testen können, sind ebenso Bestandteil der Evaluation.
Wöchentliche Auswertung der aktuellen Bugs und noch fehlender Features sind in unserem Projektfluss eingeplant worden.

> Technologien

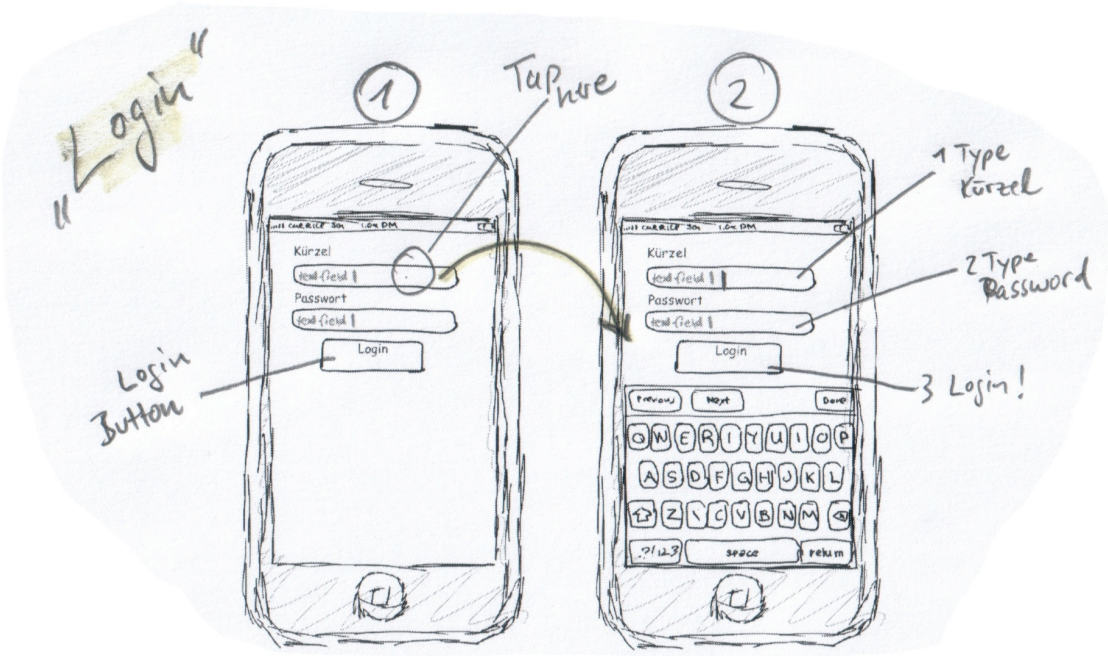
Um unseren Projektplan umzusetzen haben wir auf gängige Technologien zurückgegriffen.

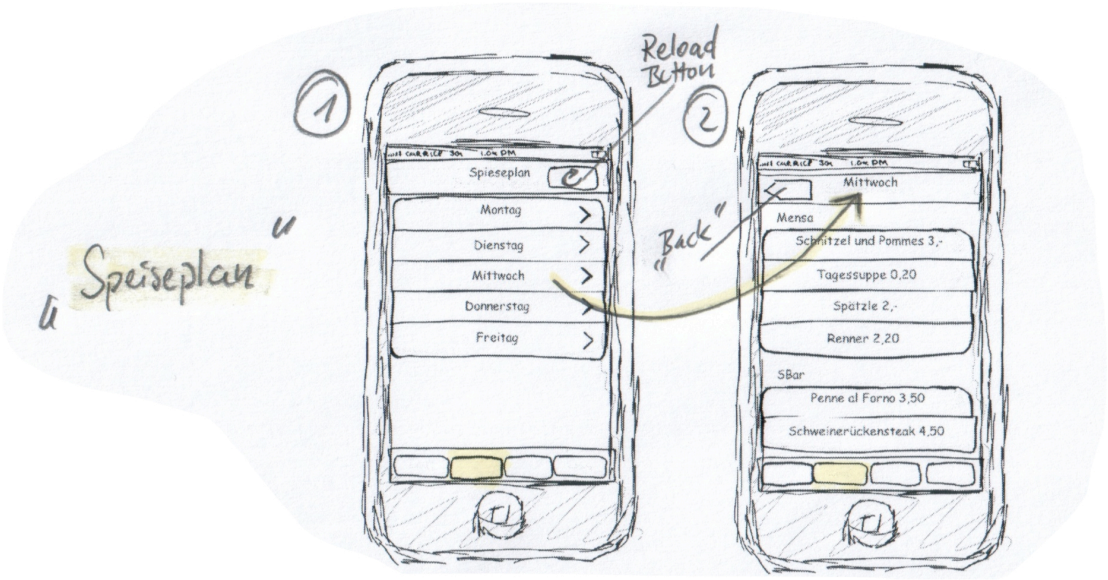
So wurde die iPhone Applikation in der Entwicklungsumgebung XCode mit der Programmiersprache Objective-C entwickelt. Hierfür wurde auch das Versionierungssystem „Subversion“ verwendet, sodass ein kollaboratives Arbeiten am Sourcecode möglich war.

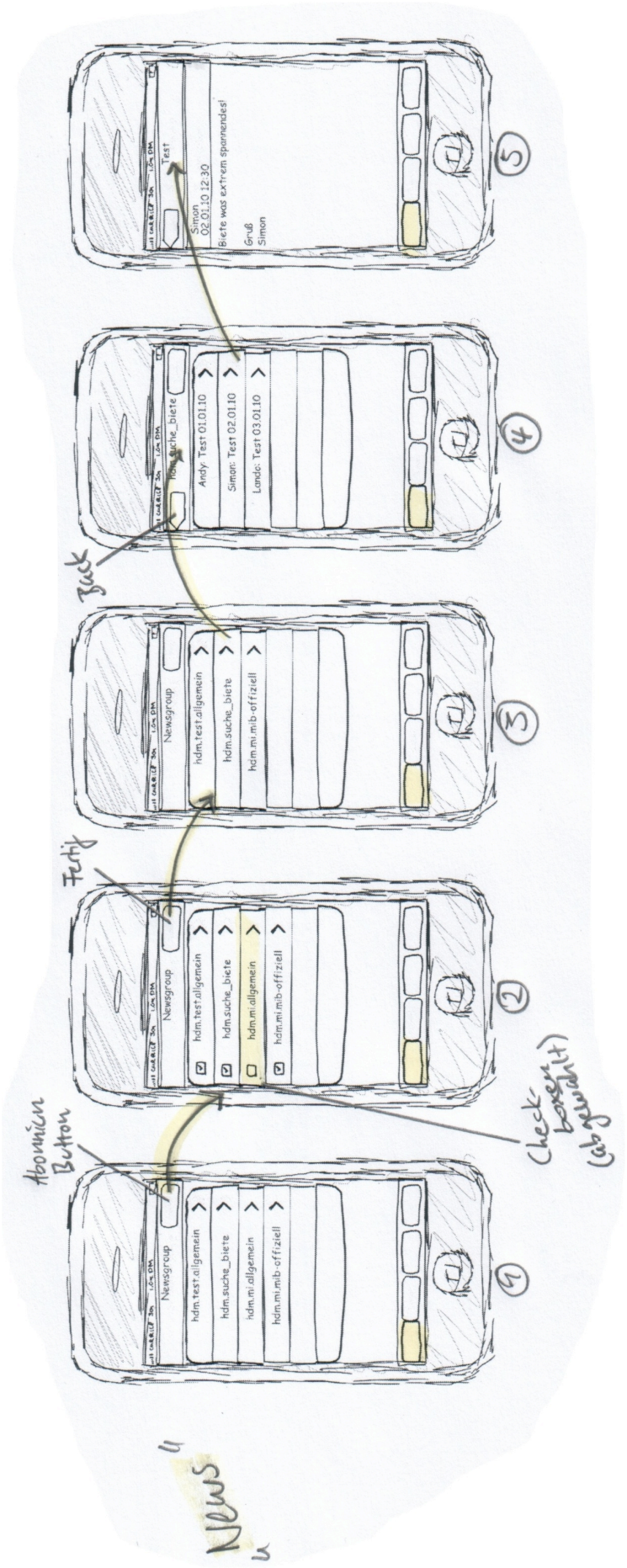
Die Entwicklung des zentralen Webbackends wurde unter Verwendung von PHP 5 realisiert. Für die Datenübertragung zwischen Webbackend und iPhone haben wir uns auf Grund der Plattformunabhängigkeit für XML entschieden.

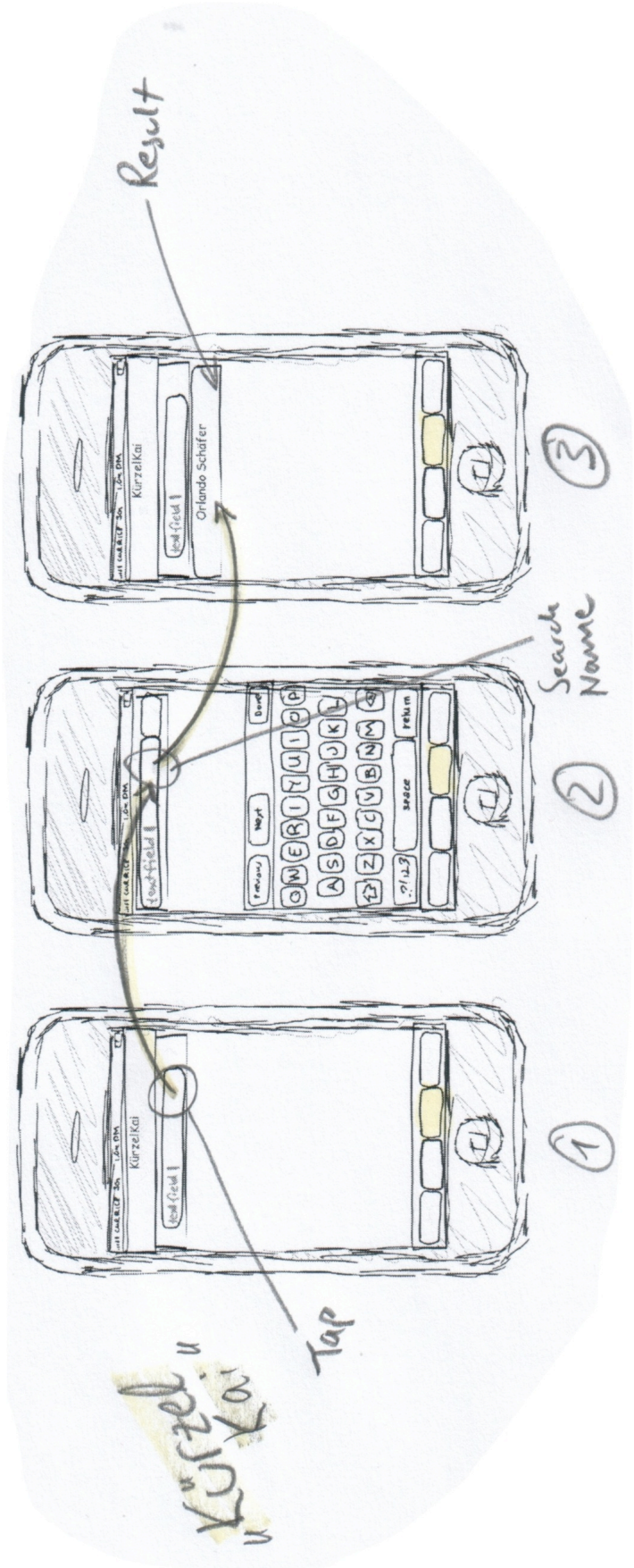
Zum Testen haben wir einerseits den iPhone Simulator, welcher Bestandteil des iPhone SDKs ist, zum anderen reale iPhones mit iOS 4.2 verwendet.

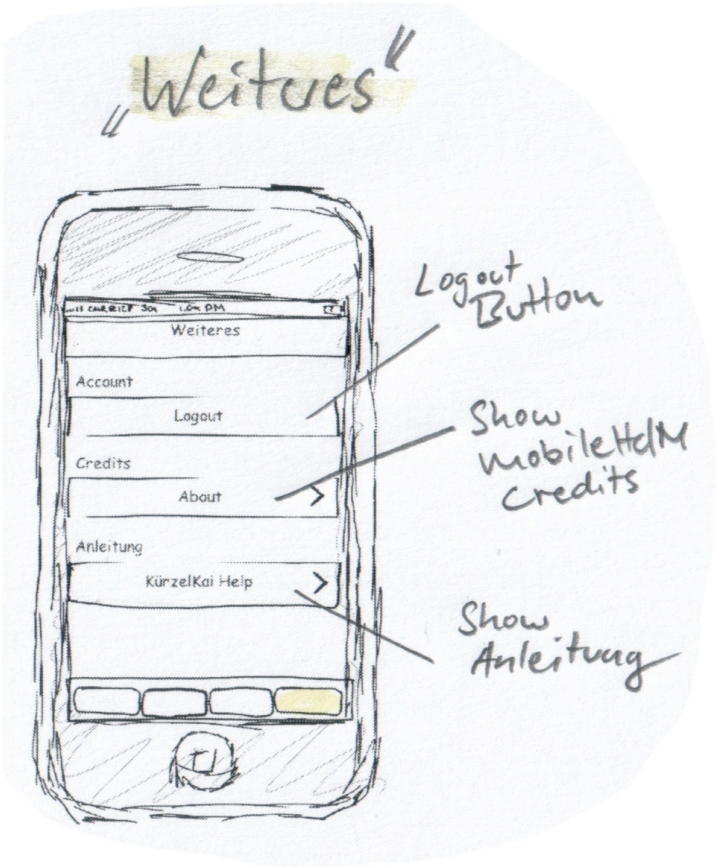
> Frühe Konzepte und Storyboard






Abonnier Button
Fertig
Zurück
Checkboxen (abgewählt)
News
Newsgroup
hdm.test.allgemein
hdm.suche_biete
hdm.mi.allgemein
hdm.mi.mib-offiziell
Andy: Test 01.01.10
Simon: Test 02.01.10
Lando: Test 03.01.10
Test
Simon
02.01.10 12:30
Biete was extrem spannendes!
Gruß
Simon


Result
Search Name
Tap
Kürzel
Kürzelkai
Orlando Schafer
1
2
3



Die Konzeptionen dienten in erster Linie dem **Storyboarding**, um zu sehen ob wir unser Drei-Klickprinzip erfüllen können. Nach dem Abonnieren der Newsgroups, was meist einmalig geschieht, ist dies auch der Fall. Das gezielte Ansteuern einer bestimmten Information ist im von uns gesetzten Rahmen schnell und übersichtlich möglich.

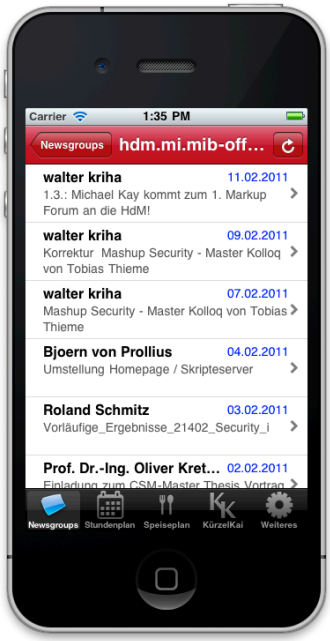
Betrachtet man das Abonnieren an sich auch als eigenständige Funktion, so ist diese ebenfalls mit 3 Klicks (bei der Auswahl vieler Newsgroups natürlich entsprechend mehr) ausführbar.

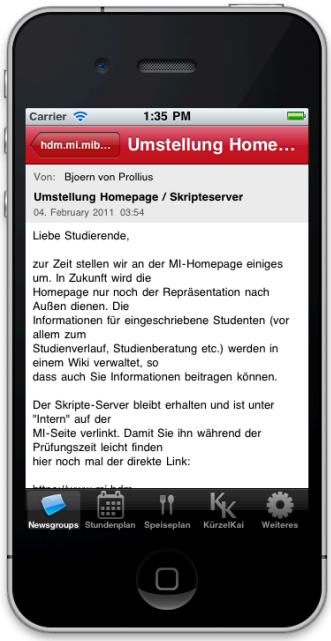
Des Weiteren wird der semantische Zusammenhang der gesamten App aus dem Storyboarding ersichtlich.

> Mockups

Login

Newsgroup


Carrier
1:35 PM
Newsgroups
hdm.mi.mib-off...
walter kriha 11.02.2011
1.3.: Michael Kay kommt zum 1. Markup Forum an die HdM!
walter kriha 09.02.2011
Korrektur Mashup Security - Master Kolloq von Tobias Thieme
walter kriha 07.02.2011
Mashup Security - Master Kolloq von Tobias Thieme
Bjoern von Prollius 04.02.2011
Umstellung Homepage / Skripteserver
Roland Schmitz 03.02.2011
Vorläufige_Ergebnisse_21402_Security_i
Prof. Dr.-Ing. Oliver Kret... 02.02.2011
Newsgroups
Stundenplan
Speiseplan
KürzelKai
Weiteres

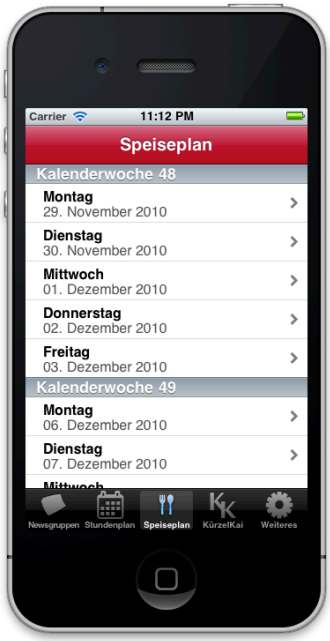

Carrier
1:35 PM
hdm.mi.mib...
Umstellung Home...
Von: Bjoern von Prollius
Umstellung Homepage / Skripteserver
04. February 2011 03:54
Liebe Studierende,
zur Zeit stellen wir an der MI-Homepage einiges um. In Zukunft wird die Homepage nur noch der Repräsentation nach Außen dienen. Die Informationen für eingeschriebene Studenten (vor allem zum Studienverlauf, Studienberatung etc.) werden in einem Wiki verwaltet, so dass auch Sie Informationen beitragen können.
Der Skripte-Server bleibt erhalten und ist unter "Intern" auf der MI-Seite verlinkt. Damit Sie ihn während der Prüfungszeit leicht finden hier noch mal der direkte Link:
Newsgroups
Stundenplan
Speiseplan
KürzelKai
Weiteres

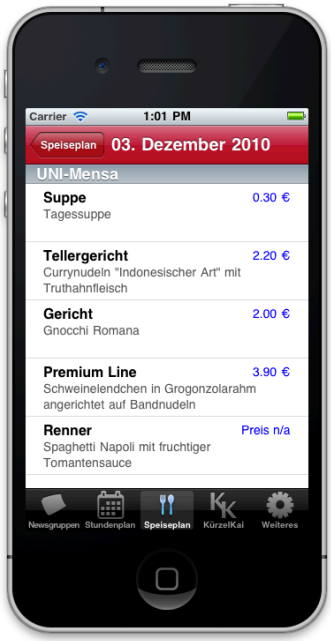
Stundeplan


Carrier
11:13 PM
Di, 30.11.
08:15
09:45
Einf. küns. Inte.
Raum 115
10:00
11:30
Einf. küns. Inte.
Raum 115
11:45
13:15
Mittagspause
14:15
15:45
User Inte. Desi.
Raum P04
16:00
17:30
User Inte. Desi.
Raum P04
Newsgruppen
Stundenplan
Speiseplan
KürzelKai
Weiteres

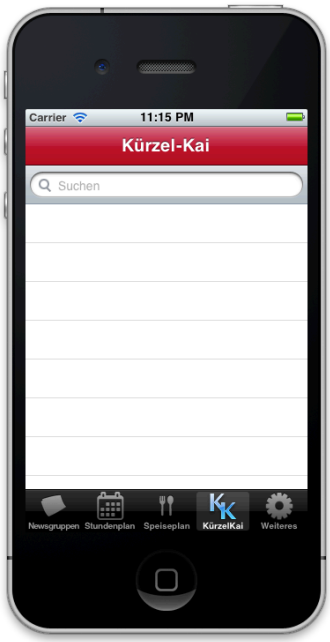

Carrier
1:02 PM
29.-05.12.
Mo (29.)
Di (30.)
Mi (01.)
Do (02.)
Fr (03.)
Mittagspause
Samstag anzeigen
Stundenplanänderungen sind rot hervorgehoben.
Newsgruppen
Stundenplan
Speiseplan
KürzelKai
Weiteres

Speiseplan





KürzelKai



Weiteres


Carrier
1:37 PM
Weiteres
Account (os019)
Logout
Credits
Credits
Anleitungen
KürzelKai
Newsgroups
Stundenplan
Speiseplan
KürzelKai
Weiteres
Carrier
1:37 PM
Weiteres
mobileHdM
Version 1.0
mobileHdM ist eine Entwicklung im Rahmen der Pflichtveranstaltung "Projekt" des Studiengangs Medieninformatik an der Hochschule der Medien im Wintersemester 2010/2011
Newsgroups
Stundenplan
Speiseplan
KürzelKai
Weiteres
Carrier
1:37 PM
Weiteres
Anleitung für den KürzelKai:
Mit Hilfe des KürzelKais ist die Suche nach einem HdM-Kürzel möglich, wenn einem nur der volle Name der gesuchten Person bekannt ist. Genauso kann aus einem Kürzel der Klarname abgefragt werden.
Geben Sie Kürzel oder Name (Umlaute möglich) in da Suchfeld ein und drücken Sie "Suche". Wenn Ihnen das vollständige Kürzel (oder der vollständige Name) nicht bekannt ist, kann mit Hilfe des *-Operators nach allen möglichen Kombinationen gesucht werden, die einen bekannten Teil enthalten.
Beispielsweise findet die Suche nach "Kai *" alle Kürzel der Studenten oder Mitarbeiter mit dem Namen Kai, die Suche nach "sl*" gibt alle Personen zurück deren Kürzel mit sl beginnt.
Newsgroups
Stundenplan
Speiseplan
KürzelKai
Weiteres

>> Entwicklung

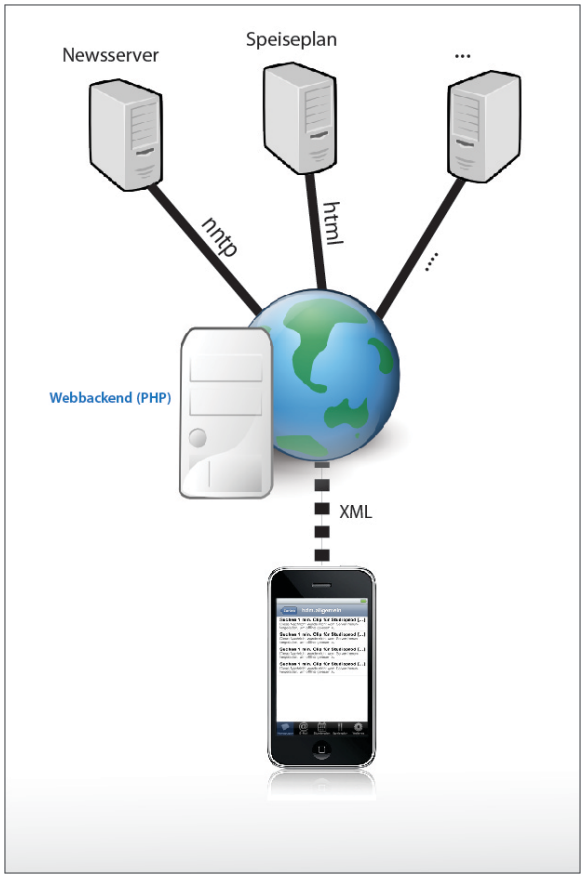
> Architekturmuster

Die Daten, welche man vom Smartphone aus abrufen möchte, liegen verteilt auf verschiedenen Servern und benutzen unterschiedliche Protokolle.

Um die Plattformunabhängigkeit der Daten zu gewährleisten werden alle Daten zunächst auf einem Webbackend gesammelt. Dieses bildet die Schnittstelle zwischen dem iPhone und den Datenquellen. Es ruft die benötigten Daten unter Verwendung der verschiedenen Protokolle ab und wandelt sie in das einheitliche XML Format um, welches dann vom iPhone und später auch anderen Smartphones abgerufen und verarbeitet werden kann.

Diese Architektur gewährleistet außerdem eine einfache Wartung des Systems. Ändert sich zum Beispiel eine Serverdomain muss keine neue Version der Applikation veröffentlicht werden sondern es reicht dies im zentralen Webbackend anzupassen.

Auch im Hinblick auf die Performance spricht vieles für diese Architektur.
Da die komplette Auswertung der Daten auf dem Webbackend geschieht muss dies nicht auf der weniger performanten Hardware der Smartphones erledigt werden.



(Bild: Architekturmuster)

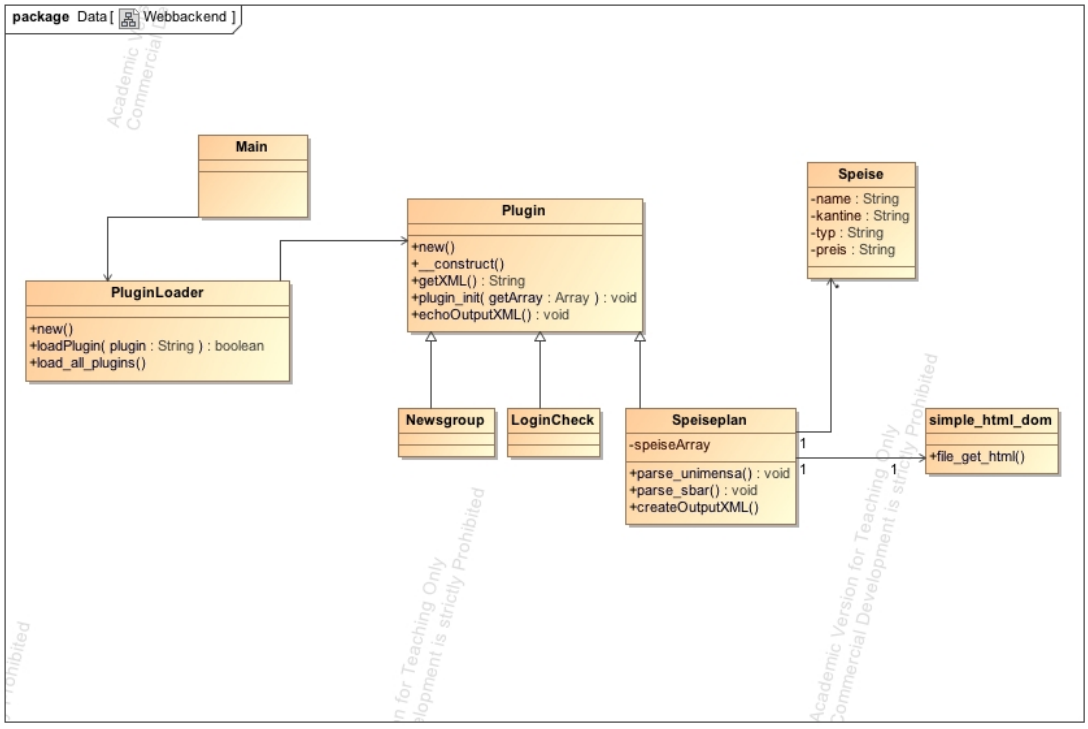
> Webbackend

Im Zuge der Überlegungen, dass diese Anwendung auch auf anderen Plattformen schnell portierbar/realisierbar sein soll, werden die Daten der einzelnen Informationsquellen und Kommunikationskanäle zunächst auf einem Webbackend gebündelt. Mögliche spätere Anwendungen für andere Systemumgebungen (z.B. für Android) können auf diese Weise schnell ihre Daten über diesen zentralen Punkt beziehen.

Das Webbackend basiert auf der Programmiersprache PHP und benötigt somit einen Webserver mit vorinstalliertem LAMP. Es dient zur Beschaffung und einfachen Weiterleitung der Daten mithilfe einer XML-Schnittstelle an das mobile Endgerät.

UML Diagramm

Das UML-Diagramm beschreibt das Plugin-System des Webbackends und zeigt beispielhaft das Plugin Speiseplan.



(Bild: UML-Diagramm)

Plugin-System

Das Webbackend ist als Plugin-System auf gebaut und somit flexibel erweiterbar um weitere Klassen (Plugins).

Erstellung eines zusätzlichen Plugins

Um ein zusätzliches Plugin für das Webbackend zu erstellen, genügt es, eine PHP-Klasse mit dem Namen des Plugins im Ordner **/plugins/** anzulegen und die eigens erstellte PHP-Klasse von der Klasse **Plugin** abzuleiten.

Der Dateiname sollte wie folgt lauten: **class.<Pluginname>.php**.
<Pluginname> wird ersetzt mit dem Namen des zu erstellenden Plugins, erstellt man z.B. ein neues Plugin mit dem Namen „NotenPlugin“, so lautet der Klassenname der PHP-Klasse **NotenPlugin** und der Dateiname **class.NotenPlugin.php** im Ordner **/plugins/**.

Nun muss das Plugin noch in der Datei **/classes/class.PluginLoader.php** registriert werden. Dazu muss der Pluginname und der URL-Name dem Attribut $plugin_array der Klasse **PluginLoader** hinzugefügt werden (in unserem Beispiel muss der Array wie folgt erweitert werden: **array(„urlname“ => „notenplugin“, „classname“ => „NotenPlugin“)**).

Der URL-Name eines Plugins ist die Umsetzung vom GET-Parameter zum Pluginnamen, so kann z.B. das Plugin mit dem URL-Parameter ?plugin=notenplugin augerufen werden, es wird jedoch die Klasse NotenFunktion geladen.

Die Grundstruktur der PHP-Klasse sollte wie folgt aussehen (anhand des Beispiels „**NotenPlugin**“):

```
class NotenPlugin extends Plugin {

	//hier stehen weitere Attribute der eigenen Klasse

	function plugin_init($getArray) {
		//hier werden die Daten geladen und in diesem Beispiel im
		Attribut
		//$this->xml zwischengespeichert
	}

	/*
	 * @Function getXML
	 *
	 * @Description: Die Funktion getXML muss von ableitenden
Plugins ueberschrieben werden, da diese Funktion als Hauptfunktion
aufgerufen wird, wenn ein Output verlangt wird
	*/

	function getXML() { return $this->xml; }

}
```

Das neu erstellte Plugin kann nun mit folgender URL geladen werden:
https://mobilehdm.mi.hdm-stuttgart.de/index.php?plugin=notenplugin

Ordnerstruktur

Beim Erstellen eines Plugins sollte aufgrund der Übersicht folgende Ordnerstruktur eingehalten werden:

/

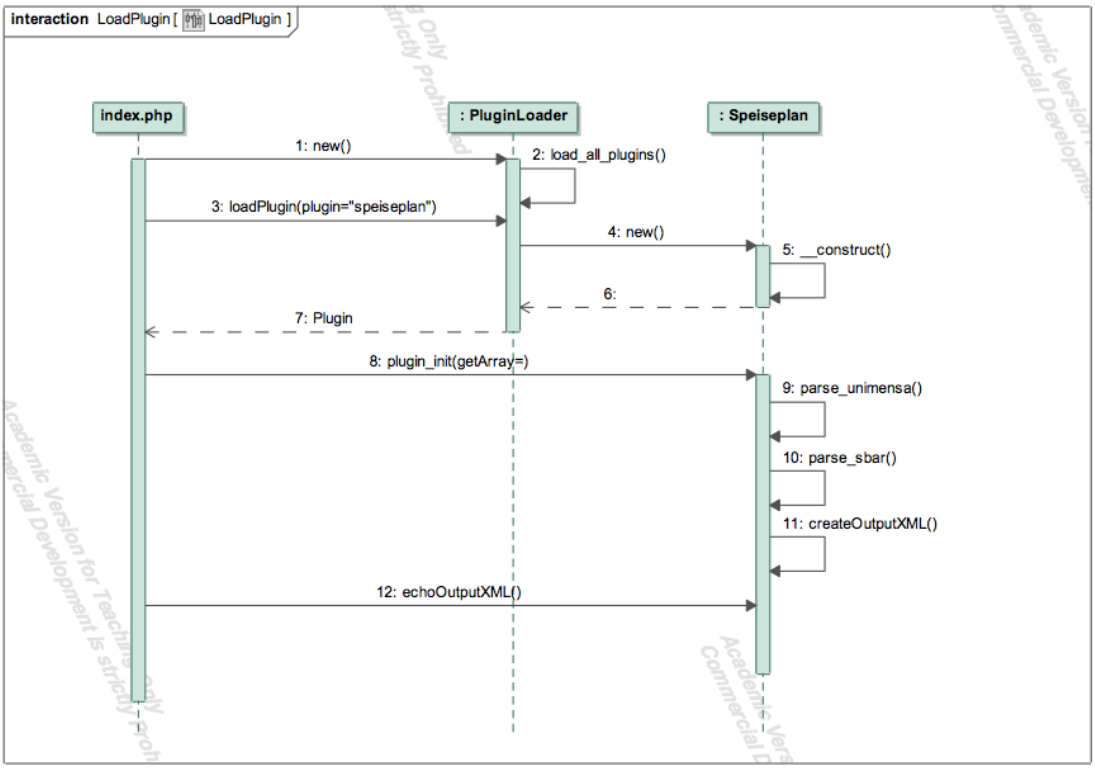
classes/ (in diesem Ordner befinden sich Systemklassen, bitte hier keine Dateien erstellen / löschen)

helper_classes/ (hier befinden sich die Hilfsklassen, die zusätzlich zur Plugin-Klasse benötigt werden.)

helper_classes/newsgroup/ (in diesem Ordner befinden sich Hilfsklassen für das Plugin „Newsgroup“, bitte für jedes Plugin aufgrund der Übersicht einen Unterordner anlegen)

plugins/ (in dieses Verzeichnis müssen die Pluginklassen gelegt werden, von dort werden die Klassen geladen und erstellt)

Ablauf



(Bild: Sequenzdiagramm (Laden des Plugins „Speiseplan“))

Im oben stehenden Bild, welches den Ablauf am Beispiel des Speiseplans darstellt, erkennt man, dass zunächst die aufgerufene index.php die Klasse PluginLoader erstellt, welche zunächst alle Plugins prüft und bei korrekter Einbindung in das Plugin-System lädt. Nun wird auf unsere vorher Erstellte PluginLoader-Klasse die Methode **loadPlugin(plugin=“Speiseplan“)** aufgerufen, die das per GET-Parameter angeforderte Plugin lädt.

Als nächster Schritt ruft die index.php Datei die Methode **plugin_init(getArray)** auf, um das Laden der Daten mithilfe des Plugins zu starten, und die Ausgabedaten in einer Variable zwischenzuspeichern. In unserem Beispiel lädt unser Speiseplan-Plugin zunächst die Daten der Unimensa (Methode: **parse_unimensa()**), dann die Daten der SBar (Methode: **parse_sbar()**) und erstellt als letzten Schritt mithilfe der Methode **createOutputXML()** die auszugebenden XML-Dateien und speichert sie im Attribut **$xml** zwischen (bis hierhin findet noch kein Output statt!).

Wenn nun unser Plugin alle Daten geladen wurden, ruft die **index.php** schließlich die Methode **echoOutputXML()** auf unserem Plugin auf, die die zwischengespeicherten Daten nun ausgibt im XML-Format.

Bisherige Plugins

Bisher wurden folgende Plugins für das Webbackend erstellt:

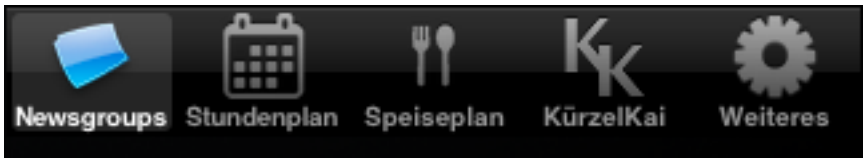
- LoginCheck (Mini-Plugin, zum Prüfen ob ein Login korrekt war)
- Newsgroup
- Speiseplan
- KuerzelKai

Eine vollständige Liste mit möglichen Zugriffen auf die Plugins per HTTP-GET liegt im Anhang bei als Excel-Tabelle.

> Die App im Allgemeinen

Die iPhone Applikation ist sehr einfach aufgebaut. Über einen **TabViewController** lassen sich am unteren Bildschirmrand über einzelne Tabs die verschiedenen Funktionen sehr leicht und übersichtlich anwählen. Diese werden über die einzelnen **ViewController** für die einzelnen Programmteile (Speiseplan, KürzelKai etc.) in der **mobileHdMAppDelegate** zentral gesteuert.

Jedes Feature besitzt seinen jeweiligen ViewController, den zugehörigen View als **.xib-Datei** und diverse Datenklassen. Wenn vom Webbackend Daten abgerufen werden, wurde hierfür noch jeweils der spezifische XMLParser implementiert (Standard **NSXMLParser**).



(Bild: Tabbar von mobileHdM)

> Login

Nach dem erstmaligen Start der Anwendung befindet sich der Benutzer im sog. Login-Bildschirm. Der Login-Bildschirm setzt sich aus dem HdM-Logo, zwei Textfeldern, in die der Benutzer seinen **Benutzernamen** und **Passwort** eintragen muss, und einem Button zur Bestätigung des Logins zusammen.

Nach der Eingabe der Benutzerdaten in die Textfelder und dem Klick (Touch) auf den Button "**Login**" werden Benutzername und Passwort in einer sicheren HTTPS Verbindung an den Server geschickt, und überprüft ob diese korrekt sind.

Bei korrekten Daten liefert der Server eine **LoginSuccessfull**-Nachricht an das iPhone zurück, dann wird der Benutzer direkt ins Hauptfenster (**TabViewController**) weitergeleitet, bei falschen Eingabedaten oder anderen Fehlern wird eine entsprechende Fehlermeldung unter dem Button in Rot angezeigt.

> Newsgroups

Die Newsgroups stellen eine zentrale Informationsquelle der HdM dar. Dort informieren Professoren, Mitarbeiter oder Studenten über Änderungen, Events und sonstige wichtige Angelegenheiten oder Informationen, die für den Studienverlauf von bedeutender Wichtigkeit sein können. „**mobileHdM**“ bietet den Studenten die Möglichkeit, je nach Studiengang die relevanten Gruppen zu abonnieren, um somit die individuell wichtigen Informationen zu herauszufiltern.

Die Newsgroups bestehen aus insgesamt drei Views, wobei jeder durch einen **ViewController** repräsentiert wird. Die Daten werden in jeder View über die entsprechende URL vom Webbackend abgerufen, geparst und lokal gespeichert. Bei den ersten beiden Views handelt es sich um **TableViews,** beim dritten um ein **ScrollView**.

Die URL-Aufrufe für die einzelnen Views (allen geht **https://mobilehdm.mi.hdm-stuttgart.de/index.php** voraus):

1. **?plugin=newsgroup**
2. **?plugin=newsgroup&show=articleList&newsgroup=XXXX&start=Y&anzahl=Z**
3. **?plugin=newsgroup&show=article&newsgroup=XXXX&artnum=Y**

Der erste View beinhaltet zum einen eine Übersicht über die abonnierten Newsgroups, zum anderen gelangt man von hier aus, durch Drücken des Abonnieren Buttons, in den „**editMode**“. Hier hat der User die Möglichkeit Newsgroups zu abonnieren oder zu entfernen. Die abonnierten Newsgroups werden für jeden User in einem Array gespeichert und in den **NSUserDefaults** abgelegt. Das Abonnieren geschieht also rein iPhone-seitig. Über die URL selbst werden immer **alle** Newsgroups abgerufen.

Wählt man nun, nach dem Abonnieren, eine Newsgroup aus, gelangt man in den zweiten View, in welchem man zunächst die 20 neusten Themen angezeigt bekommt. Am unteren Ende dieses TableViews befindet sich eine Zelle mit dem Namen „**Weitere Nachrichten anzeigen**“. Durch die Auswahl dieser Zelle wird ein **Counter**, welcher die Anzahl der angezeigten Themen in der Newsgroup repräsentiert, um **20** erhöht (s.o. „**anzahl**“ in der zweiten URL).
Danach wird die URL zum Newsgroup-Plugin auf dem Webbackend mit dem neuen Nachrichten-Counter aufgerufen und die nächsten 20 Themen in den TableView geladen.

Oben rechts in der **NavigationBar** befindet sich ein **Reload** Button. Mit diesem kann geprüft werden, ob es neue Themen gibt und falls ja, werden diese vom Webbackend abgerufen.

Durch einen Tap auf eines der Themen gelangt man in den dritten und letzten View der Newsgroups. Hier wird der Inhalt der ausgewählten Nachricht angezeigt, nachdem der entsprechende Artikelindex abermals über die URL mitgeliefert wurde (s.o. „**artnum**“ in der dritten URL).

Bereits angewählte Listen und Nachrichten werden ebenso in den **NSUserDefaults** gespeichert, sodass diese nicht jedesmal neu geladen werden müssen. Das Caching erhöht zudem die Geschwindigkeit der Navigation und verbessert die Benutzerfreundlichkeit.

> Speiseplan

Mithilfe der „**mobileHdM**“ Anwendung haben Studenten einen mobilen Zugriff auf sämtliche Speisepläne der SBar oder der Uni-Mensa. Somit können Studenten schon im Voraus entscheiden welche Speise sie zu sich nehmen wollen und zu welchem Preis. Damit spart man sich ungewollte Überraschungen, kann seine Pausen besser planen und spart sich eventuell unnötig lange Märsche zur Uni-Mensa.

Der Speiseplan wurde relativ einfach, wie auch schon die Newsgroups, mittels zwei **TableViewControllers** realisiert.

Zunächst werden die Daten über die URL **https://mobilehdm.mi.hdm-stuttgart.de/index.php?plugin=speiseplan&kw=xxxx** vom Backend abgerufen. Dem Parameter „**kw**“ wird der aktuelle Timestamp mitgeliefert. Über diesen wird die aktuelle Kalenderwoche zurückgeliefert, sodass auf der **Unimensa** Seite auf die richtige Speiseplanwoche zugegriffen wird. Sollte Wochenende sein (Samstag oder Sonntag), wird

der Timestamp der kommenden Woche übergeben, sodass die nächsten Gerichte vor Wochenbeginn bereits abrufbar sind.

Von der **S-Bar** Seite werden immer alle Gerichte zurückgeliefert, da meist nur eine Woche, nämlich die aktuelle, vorhanden ist.

Der zuletzt korrekt übermittelte Speiseplan bleibt in den **NSUserDefaults** gespeichert und wird im Falle, dass keine Internetverbindung besteht oder anderweitige Fehler auftreten für die Aufbereitung der Daten verwendet.

Über den **MenuXMLParser** wird das zurückgelieferte XML geparst und die Gerichte in Form eines **NSMutableDictionary** (der aus Date-Object Einträgen besteht) gespeichert.

Dieses Dictionary ist nach Kalenderwochen geordnet. Jede Kalenderwoche stellt eine eigene **Table Section** dar und beinhaltet die einzelnen Daten und Wochentage, für die es Gerichte gibt. Tage, die in der Vergangenheit liegen werden ausgeblendet – Der aktuelle Tag wird grau hinterlegt.
Über einen **Refresh** Button in der **NavigationBar** können die Daten neu geparst und geladen werden.

Nach einem Tap auf den entsprechenden Tag öffnet sich ein zweiter TableView. Hier wird aus dem Dictionary das entsprechende Date-Object herausgeholt und an jenen TableView übergeben. Das Date-Object beinhaltet wiederum ein **NSMutableArray**, welches die einzelnen Gerichte des Tages gespeichert hat.

Was folgt ist eine einfache tabellarische Aufbereitung der Speisen mit den Attributen Gerichttyp, Gerichtname und Preis. Die Table Sections werden nun von den zwei Kantinen, Unimensa und S-Bar, repräsentiert.

Sollte in der Zukunft eine neue Mensa hinzukommen (was mit dem Neubau passieren könnte) ist dies iPhone-seitig sehr einfach zu bewerkstelligen. Es muss lediglich in der Methode „**viewDidLoad()**“ des **MealViewControllers** folgende Zeile mit dem Kantinennamen (hier als Beispiel „**SBar2**“) erweitert werden:

```
// If there will be more canteens in the future, add them here

self.canteenNames = [[NSArray alloc] initWithObjects:@"unimensa",
@"SBar", @“SBar2”, nil];
```

Die einfache Erweiterbarkeit auf Seiten des Backends wurde im entsprechenden Kapitel bereits vorgestellt. Hier wäre nur das Plugin **Speiseplan** um die Methode **parse_sbar2()** zu erweitern, sodass auch die Gerichte jener Kantinenseite korrekt geparst werden können.

> Stundenplan

Der Stundenplan beinhaltet für jeden HdM-Studenten seine persönlich eingetragenen Veranstaltungen für das jeweilige Semester. Sie werden geordnet als Wochenplan dargestellt. Dieser enthält Informationen über die Semesterwochenstunden, die ECTS-Punkte, Termine und Räume der einzelnen Veranstaltungen. Zudem werden Verlegungen, ob zeitlich oder räumlich, und Ausfälle vermerkt.

Der Stundenplan wurde bereits im Sommersemester 2010 als Projekt einiger Studenten umgesetzt, somit mussten wir lediglich einen **UIWebView** erstellen und diesen Anweisen, die URL **https://www.hdm-stuttgart.de/msp** aufzurufen.

> Kürzelkai

Mithilfe der KürzelKais lässt sich der HdM **LDAP-Server** nach einem Kürzel durchsuchen. Dazu wurden zwei **UITableViews** erstellt, eine, die die aktuellen Suchergebnisse anzeigt und eine, die die letzten gesuchten Personen beinhaltet.

Nach Abschicken des zu suchenden Kürzels auf dem iPhone, wird auf dem LDAP-Server eine Suchanfrage gestartet, die alle Personen zurückliefert, auf die das Suchmuster zutrifft. Diese Personenliste wird im XML-Format zurück an das iPhone geliefert. Als **Wildcard** kann das „*“ Symbol verwendet werden.
Erhält man eine gültige Personenliste vom Server zurück, wird diese zunächst im **UISearchBarTableView** angezeigt, und dann in die **TableView** übernommen, die die letzten Suchergebnisse darstellt.

> Fehlerhandling

Um es den Entwicklern und Benutzern einfacher zu machen, wurde ein erweitertes Fehlerhandling mit in das Webbackend und in die Anwendung eingebaut. Wenn im Plugin während eines Prozesses Fehler auftreten, werden diese in einer standardisierte XML-Datei zurück an das iPhone geliefert, dieses muss dann lediglich bei jedem Aufruf an das Webbackend überprüfen, ob einer dieser Fehler aufgetreten sind, oder ob richtige Daten zurückgegeben wurden.

Diese Methode verschafft uns Entwicklern den klaren Vorteil, dass der Benutzer, wenn ein Fehler auftritt, sich einfach an den Entwickler wenden kann, und genaustens erklären, welcher Fehlercode und welche Fehlermeldung er erhalten hat. Der Entwickler schaut dann lediglich in einer Tabelle nach, in der alle Fehler und Codestelle aufgelistet sind, und findet sofort die Stelle, an der der Fehler aufgetreten ist.

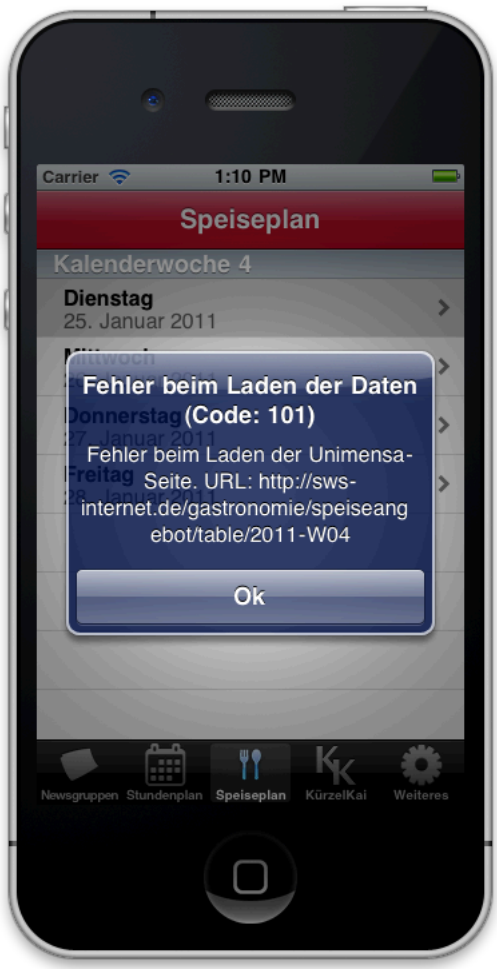
Die Fehler haben folgendes XML-Format (am Beispiel eines falsch geladenen Plugins):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mobileHdM SYSTEM "general.dtd">
<mobileHdM>
	<returnType>
		<![CDATA[PluginLoaderError]]>
	</returnType>
	<content>
		<message>
			<![CDATA[Dieses Plugin ist nicht aktiv.]]>
		</message>
		<code>
			<![CDATA[2]]>
		</code>
		<file>
			<![CDATA[/var/www/classes/class.PluginLoader.php]]>
		</file>
		<line>
			<![CDATA[83]]>
		</line>
	</content>
</mobileHdM>
```

Wie man am vorherigen Code erkennen kann, besteht ein Fehler aus mehreren Feldern, die je nach belieben in der iPhone Anwendung ausgewertet und ausgegeben werden können.

Message: Gibt die ausführliche Fehlermeldung wieder.
Code: Der Fehlercode. Anhand dieses Fehlercodes kann der Fehler schnell im eigentlichen Quellcode gefunden werden.
File: Die Datei, in der der Fehler aufgetreten ist.
Line: Die Zeile, in der der Fehler aufgetreten ist.

Eine **Beispielfehlermeldung** auf dem iPhone sieht dann wie folgt aus:



(Bild: Fehlermeldung für den Speiseplan)

>> Fazit

mobileHdM ist der erste Schritt.

Wir haben die Grundlage für viele denkbare aufbauende Projekte geschaffen. Sei es der mobileHoRST, das fehlende Notenplugin oder eine Social Web Komponente. Durch das Webbackend ist eine Lösung für weitere App-Realisierungen auf anderen Plattformen kreiert worden, die sicher dieselben und auch viele andere Funktionen, die die jetzige iPhone App nicht beinhaltet, umfassen könnten.

mobileHdM hat im ersten Ansatz sicher das erreicht, was wir ursprünglich auch wollten.

Studenten informieren sich schnell und einfach über ihr Mobiltelefon und behalten den Überblick – egal wo sie sich auch zu welchem Zeitpunkt auch immer befinden. Dies hat die Praxis während der Entwicklung immer wieder gezeigt. Die Vorfreude wurde während der gesamten Entstehungsphse immer größer, denn das Wissen, dass tatsächlich Nutzen daraus gezogen wird, motivierte das gesamte Team.

Für alle Mitglieder des Projektteams war es eine tolle Erfahrung und mit Sicherheit ein nützlicher und erfolgreicher Einstieg in die Welt der mobilen Medien.

Wir wünschen allen viel Spaß mit **mobileHdM**!

>> Anhang

> Webbackend API

#	Plugin	URL	Beschreibung	Anmerkung
1	Speiseplan	?plugin=speiseplan	Alle Speisen der Sbar + die Speisen der Unimensa der aktuellen Kalenderwoche	
2	Speiseplan	?plugin=speiseplan&kw=1289817113	Alles Speisen der Sbar + die Speisen der Unimensa der KW 46	Der Parameter Kalenderwoche muss im Format eines Zeitstempels (Timestamps) angegeben werden. Dieser stellt die vergangene Zeit (in s) seit dem 01.01.1970 dar. Das PHP-Skript ermittelt daraus automatisch die Kalenderwoche.
3	Newsgroup	?plugin=newsgroup	Alle Newsgruppen auflisten	
4	Newsgroup	?plugin=newsgroup&show=articleList &newsgroup=hdm.suche_biete	20 Artikel einer Newsgruppe „hdm.suche_biete“ anzeigen, beginnend beim 1. Artikel	
5	Newsgroup	?plugin=newsgroup&show=articleList &newsgroup=hdm.suche_biete&start =20&anzahl=35	35 Artikel einer Newsgruppe „hdm.suche_biete“ anzeigen, beginnend beim 20. Artikel	
6	Newsgroup	?plugin=newsgroup&show=article&ne wsgroup=hdm.suche_biete&artnum=1 118	Artikel 1118 der Newsgruppe "hdm.suche_biete" anzeigen	Die Artikelnummer muss in der Gruppe "hdm.suche_biete" gepostet sein
7	KuerzelKai	?plugin=kuerzelkai&search=sl064	Suche Personen im LDAP mit dem Kürzel sl064	
8	KuerzelKai	?plugin=kuerzelkai&search=sl*	Suche Personen im LDAP, die mit den Buchstaben sl anfangen.	* kann bei der Suche nach einem Kürzel als Wildcard verwendet werden
9	LoginCheck	?plugin=logincheck	Liefert eine einfache XML-Datei zurück	

> Webbackend Errorliste

#	Klasse	Name	Code	Meldung	Beschreibung	Lösung
1	class.PluginLoader.php	PluginLoaderError	1	Kein gültiges Plugin in der URL angegeben.	Dieser Fehler tritt auf, wenn der HTTP-GET Parameter "plugin" fehlt oder leer ist.	HTTP-GET Parameter "plugin" auf den gewünschten Pluginnamen setzen
2	class.PluginLoader.php	PluginLoaderError	2	Dieses Plugin ist nicht aktiv.	Das Plugin ist nicht aktiv geschalten.	Beheben lässt sich der Fehler mit einem richtigen Eintrag im Array-Attribut "plugin_array" der Klasse class.PluginLoader.php. **Format:** array('urlname' =>'<URLName>', 'classname' => '<Klassenname>'). **Erläuterung:** <URLName> ist der Name, der als HTTP-GET Parameter übergeben wird (z.B. ?plugin=<URLName>), Klassenname ist der Name der PHP-Klasse
3	class.PluginLoader.php	PluginLoaderError	3	Klassenname des Plugin wurde nicht gefunden ($plugin_array richtig gesetzt?).	Das Array-Attribut "plugin_array" wurde falsch gesetzt.	s. Fehler #2
4	class.PluginLoader.php	PluginLoaderError	4	Die Klasse <Klassenname> wurde nicht gefunden.	Die Klasse des Plugins wurde nicht gefunden.	Der Klassenname der Plugin-Klasse muss so lauten, wie im "plugin_array" Attribut angegeben.
5	class.Speiseplan.php	SpeiseplanError	101	Fehler beim Laden der Unimensa-Seite. URL: <URL>	Die Unimensa Seite konnte nicht erreicht werden (geänderte URL?)	Unimensa verklagen
6	class.Speiseplan.php	SpeiseplanError	102	Fehler beim Parsen der Unimensa-Seite.	Die Unimensa Seite konnte nicht richtig geparsed werden.	Der HTML Quellcode wurde modifiziert, somit muss der HTML-Parser angepasst werden auf den neuen Quellcode.

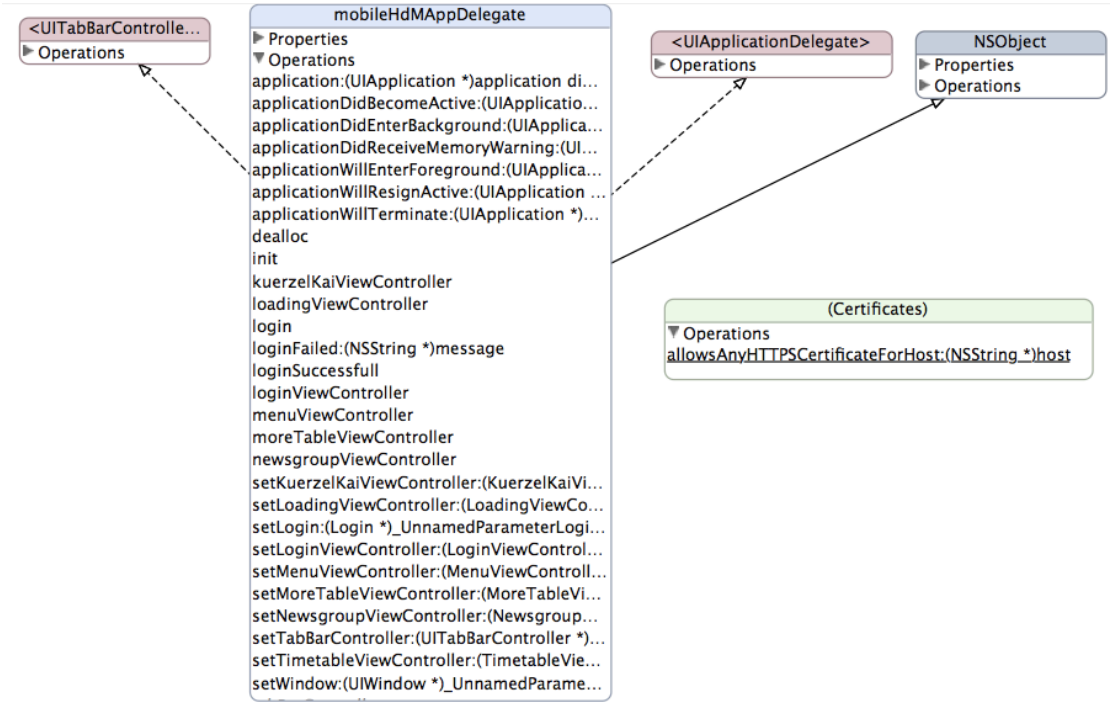
12	class.Speiseplan.php	SpeiseplanError	108	Fehler beim Parsen der SBar-Seite.	Die Unimensa Seite konnte nicht richtig geparsed werden. Der HTML Quellcode wurde modifiziert, somit muss der HTML-Parser angepasst werden auf den neuen Quellcode.	Der HTML Quellcode wurde modifiziert, somit muss der HTML-Parser angepasst werden auf den neuen Quellcode.
13	class.Newsgroup.php	NewsgroupError	201	Benötigten Parameter 'newsgroup' nicht angegeben.	Der Parameter "newsgroup" wurde nicht angegeben.	HTTP-GET Parameter "newsgroup" auf die gewünschte Newsgruppe setzen
14	class.Newsgroup.php	NewsgroupError	202	Die Newsgruppe mit dem Namen <Newsgruppe> wurde nicht gefunden.	Es wurde eine Newsgruppe angegeben, die nicht existiert.	Newsgruppen-Liste abrufen mit ?plugin=newsgroup und passende Newsgruppe heraussuchen
15	class.Newsgroup.php	NewsgroupError	203	Benötigten Parameter 'artnum' nicht angegeben.	Der Parameter "artnum" zum Abrufen eines bestimmten Artikels wurde nicht mit angegeben	Artikelliste abrufen mit ?plugin=newsgroup&show=articleList&newsgroup=<Newsgruppe> und entsprechende Nachricht heraussuchen
16	class.Newsgroup.php	NewsgroupError	204	Der Artikel mit der ArtikelNummer <ArtikelNummer> wurde nicht gefunden.	Der Artikel mit der Nummer <ArtikelNummer> wurde nicht in der angegebenen Newgruppe gefunden.	s. #16

7	class.Speiseplan.php	SpeiseplanError	103	Fehler beim Parsen der Unimensa-Seite. Contenttabelle nicht gefunden.	Die Tabelle mit den Speisen als Inhalt wurde nichte gefunden.	s. #6
8	class.Speiseplan.php	SpeiseplanError	104	Attribut $unimensa_domain nicht gesetzt.	Das Attribut "unimensa_domain" der Klasse class.Speiseplan.php ist nicht richtig gesetzt	Attribut richtig setzen auf die Domain der Unimensa.
9	class.Speiseplan.php	SpeiseplanError	105	Ungültiger Parameterwert	Es wurde ein ungültiges Kalenderjahr oder eine ungültige Kalenderwoche übergeben	PHP-Code ausbessern, eigentlich sollte dieser Fehler nicht auftreten
10	class.Speiseplan.php	SpeiseplanError	106	Fehler beim Parsen der Unimensa-Seite: Datumsformat falsch.	Das Datumsformat auf der Unimensa-Seite hat sich geändert.	HTML-Parser anpassen auf das neue Datumsformat
11	class.Speiseplan.php	SpeiseplanError	107	Fehler beim Laden der SBar-Seite. URL: <URL>	Die SBar Seite konnte nicht erreicht werden (geänderte URL?)	SBar verklagen oder URL-Attribute "sbar_domain" und "sbar_url" anpassen

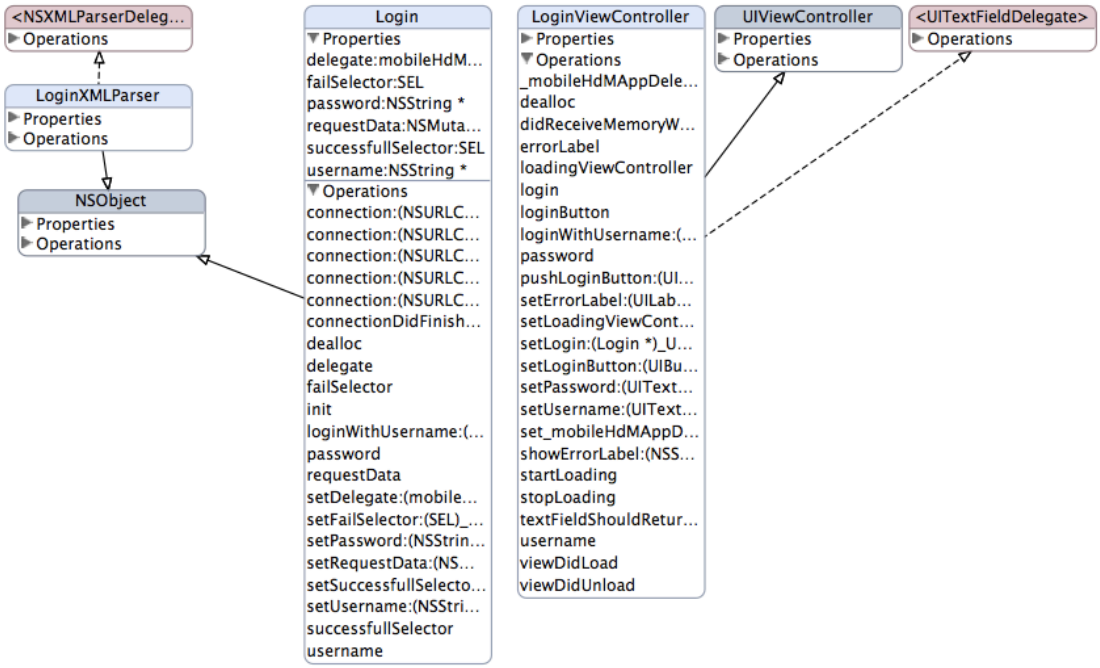
17	class.Newsgroup.php	NewsgroupError	205	Anmeldung am Newsserver ist fehlgeschlagen.	Es ist ein Fehler beim Verbinden mit dem Newsserver aufgetreten.	Konfiguration des Plugins class.Newsgroup.php überprüfen oder testen ob der Newsserver online ist
18	class.Newsgroup.php	NewsgroupError	206	class.Newsgroup.php :: getArticlesForGroup(): Keine Gruppe angegeben.	Interner Fehler	Interner Fehler des Plugins class.Newsgroup.php, Plugin prüfen
19	class.KuerzelKai.php	KuerzelKaiError	301	Ungültiger Parameter angegeben.	Der Parameter 'search' ist ungültig	Es sind nur die Zeichen a-Z, ä , ü, ö, - und Leerzeichen erlaubt.
20	class.KuerzelKai.php	KuerzelKaiError	302	Konnte keine Verbindung zum LDAP-Server herstellen.	Es konnte keine Verbindung zum LDAP Server hergestellt werden.	Prüfen ob LDAP Server vorhanden und Plugin richtig konfiguriert

> UML Diagramme „mobileHdM“

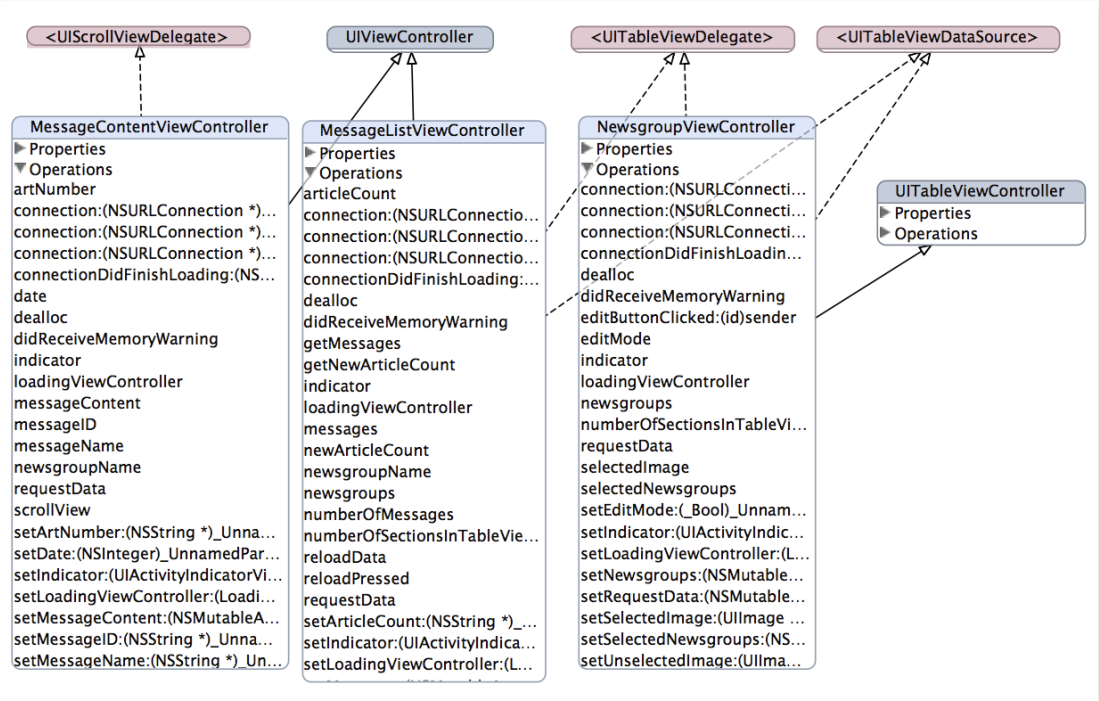
AppDelegate

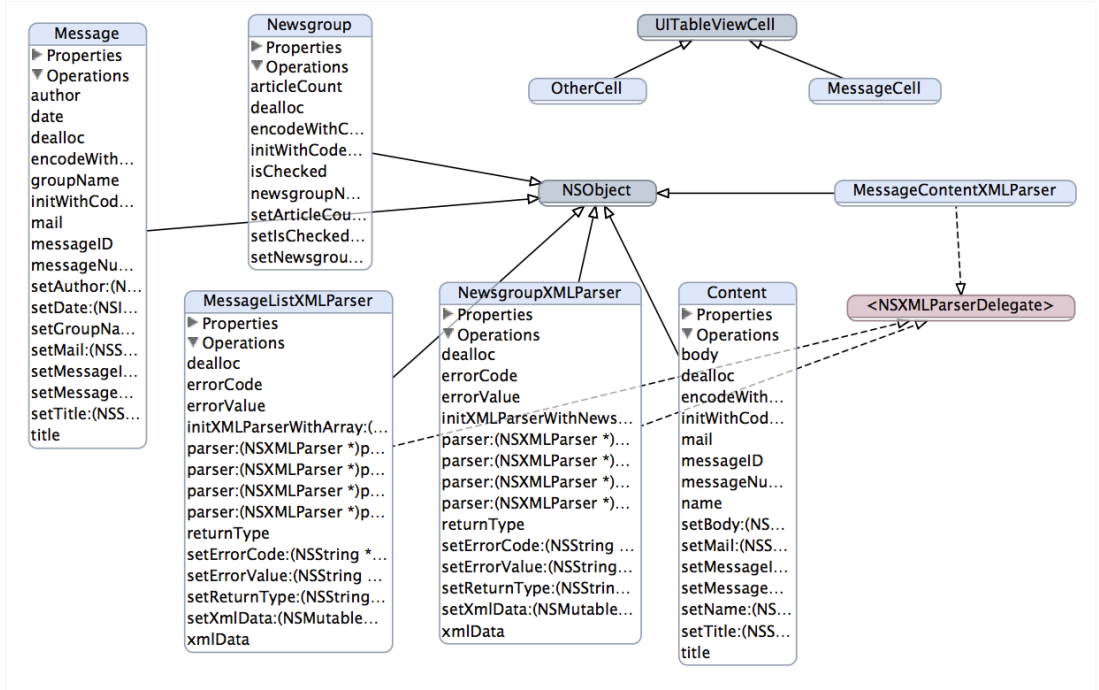


Login

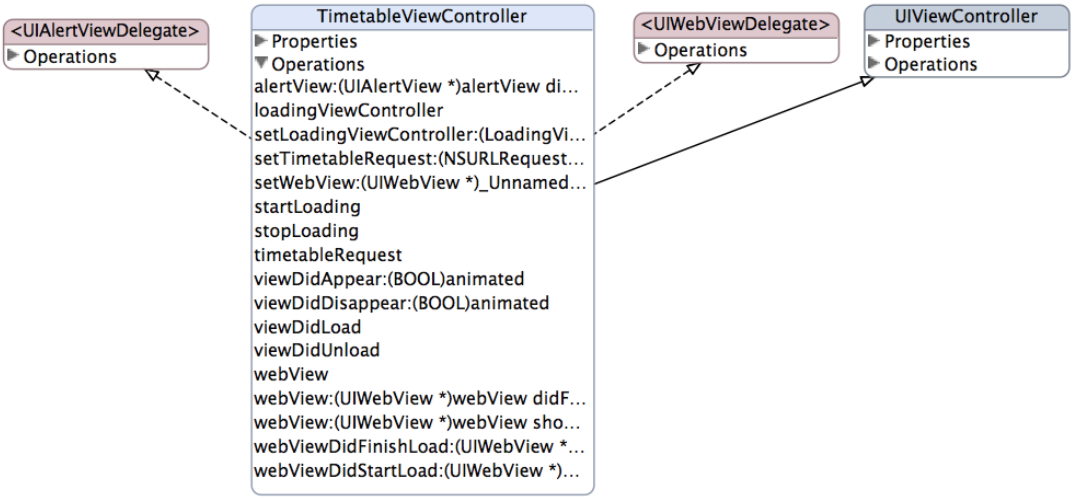


Newsgroups

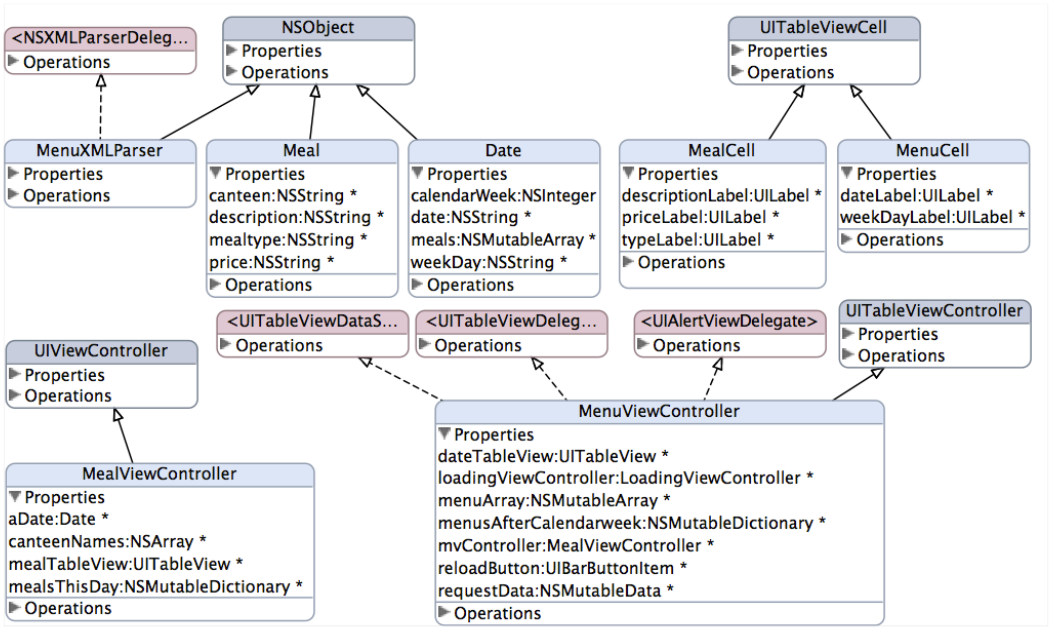

<UIScrollViewDelegate>
UIViewController
<UITableViewDelegate>
<UITableViewDataSource>
MessageContentViewController
Properties
Operations
artNumber
connection:(NSURLConnection *)...
connection:(NSURLConnection *)...
connection:(NSURLConnection *)...
connectionDidFinishLoading:(NS...
date
dealloc
didReceiveMemoryWarning
indicator
loadingViewController
messageContent
messageID
messageName
newsgroupName
requestData
scrollView
setArtNumber:(NSString *)_Unna...
setDate:(NSInteger)_UnnamedPar...
setIndicator:(UIActivityIndicatorVi...
setLoadingViewController:(Loadi...
setMessageContent:(NSMutableA...
setMessageID:(NSString *)_Unna...
setMessageName:(NSString *)_Un...
MessageListViewController
Properties
Operations
articleCount
connection:(NSURLConnectio...
connection:(NSURLConnectio...
connection:(NSURLConnectio...
connectionDidFinishLoading:...
dealloc
didReceiveMemoryWarning
getMessages
getNewArticleCount
indicator
loadingViewController
messages
newArticleCount
newsgroupName
newsgroups
numberOfMessages
numberOfSectionsInTableVie...
reloadData
reloadPressed
requestData
setArticleCount:(NSString *)_...
setIndicator:(UIActivityIndica...
setLoadingViewController:(L...
NewsgroupViewController
Properties
Operations
connection:(NSURLConnecti...
connection:(NSURLConnecti...
connection:(NSURLConnecti...
connectionDidFinishLoadin...
dealloc
didReceiveMemoryWarning
editButtonClicked:(id)sender
editMode
indicator
loadingViewController
newsgroups
numberOfSectionsInTableVi...
requestData
selectedImage
selectedNewsgroups
setEditMode:(_Bool)_Unnam...
setIndicator:(UIActivityIndic...
setLoadingViewController:(L...
setNewsgroups:(NSMutable...
setRequestData:(NSMutable...
setSelectedImage:(UIImage ...
setSelectedNewsgroups:(NS...
setUnselectedImage:(UIIma...
UITableViewController
Properties
Operations


Message
Properties
Operations
author
date
dealloc
encodeWith...
groupName
initWithCod...
mail
messageID
messageNu...
setAuthor:(N...
setDate:(NSI...
setGroupNa...
setMail:(NSS...
setMessageI...
setMessage...
setTitle:(NSS...
title
Newsgroup
Properties
Operations
articleCount
dealloc
encodeWithC...
initWithCode...
isChecked
newsgroupN...
setArticleCou...
setIsChecked...
setNewsgrou...
UITableViewCell
OtherCell
MessageCell
NSObject
MessageContentXMLParser
MessageListXMLParser
Properties
Operations
dealloc
errorCode
errorValue
initXMLParserWithArray:(...
parser:(NSXMLParser *)p...
parser:(NSXMLParser *)p...
parser:(NSXMLParser *)p...
parser:(NSXMLParser *)p...
returnType
setErrorCode:(NSString *...
setErrorValue:(NSString ...
setReturnType:(NSString...
setXmlData:(NSMutable...
xmlData
NewsgroupXMLParser
Properties
Operations
dealloc
errorCode
errorValue
initXMLParserWithNews...
parser:(NSXMLParser *)...
parser:(NSXMLParser *)...
parser:(NSXMLParser *)...
parser:(NSXMLParser *)...
returnType
setErrorCode:(NSString ...
setErrorValue:(NSString...
setReturnType:(NSStrin...
setXmlData:(NSMutable...
xmlData
Content
Properties
Operations
body
dealloc
encodeWith...
initWithCod...
mail
messageID
messageNu...
name
setBody:(NS...
setMail:(NSS...
setMessageI...
setMessage...
setName:(NS...
setTitle:(NSS...
title
<NSXMLParserDelegate>

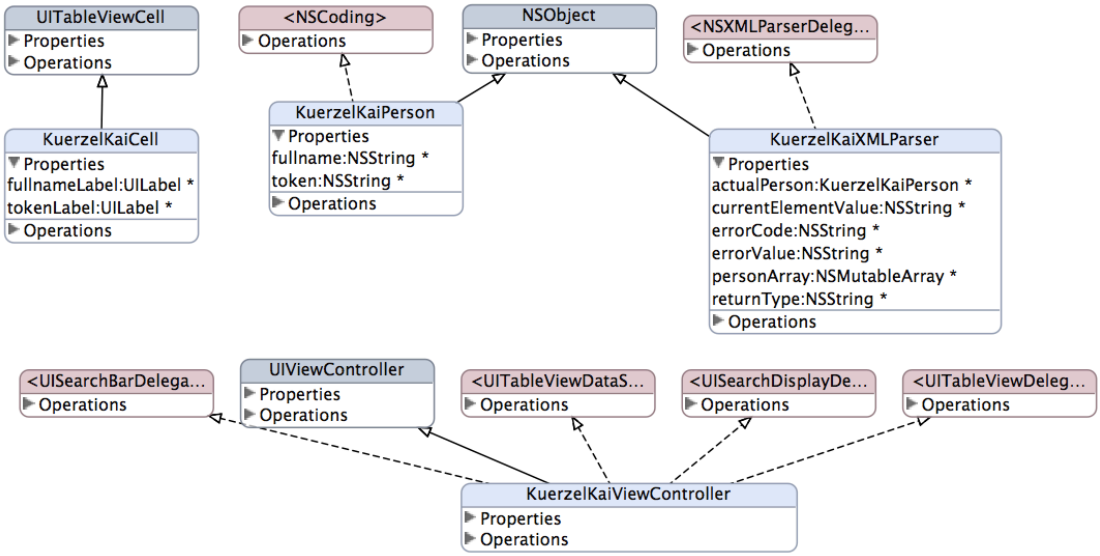
Stundenplan



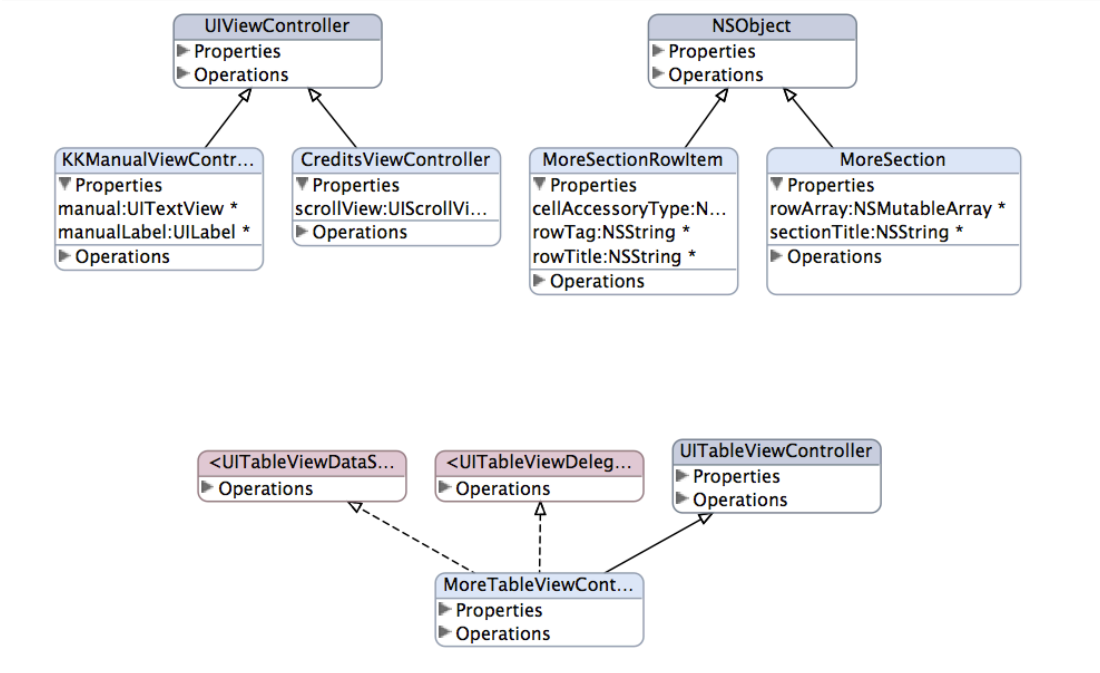
Speiseplan



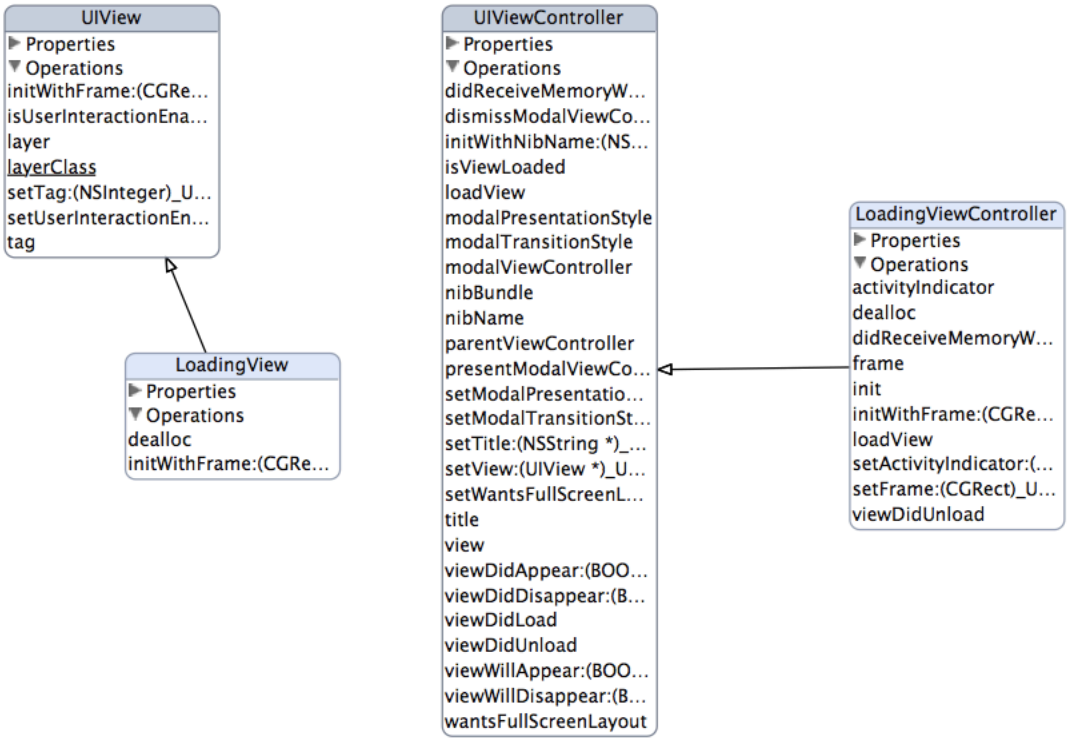
KürzelKai



„Weiteres“-Sektion



Loading


UIView
Properties
Operations
initWithFrame:(CGRe...
isUserInteractionEna...
layer
layerClass
setTag:(NSInteger)_U...
setUserInteractionEn...
tag
LoadingView
Properties
Operations
dealloc
initWithFrame:(CGRe...
UIViewController
Properties
Operations
didReceiveMemoryW...
dismissModalViewCo...
initWithNibName:(NS...
isViewLoaded
loadView
modalPresentationStyle
modalTransitionStyle
modalViewController
nibBundle
nibName
parentViewController
presentModalViewCo...
setModalPresentatio...
setModalTransitionSt...
setTitle:(NSString *)_...
setView:(UIView *)_U...
setWantsFullScreenL...
title
view
viewDidAppear:(BOO...
viewDidDisappear:(B...
viewDidLoad
viewDidUnload
viewWillAppear:(BOO...
viewWillDisappear:(B...
wantsFullScreenLayout
LoadingViewController
Properties
Operations
activityIndicator
dealloc
didReceiveMemoryW...
frame
init
initWithFrame:(CGRe...
loadView
setActivityIndicator:(...
setFrame:(CGRect)_U...
viewDidUnload