
Hochschule für Telekommunikation Leipzig
University of Applied Sciences

# ENTWICKLUNG EINER HFTL-APP
# DOKUMENTATION

Studienmodul *Software-Engineering*
der Hochschule für Telekommunikation
Leipzig

**Projektarbeit - Softwareentwicklung**

vorgelegt von

**BKMI Matrikel 13**

23.04.2015

**Dozent:** Profn. Dr.-Ing. Sabine Wieland

# INHALTSVERZEICHNIS

# 1 Einführung

## 1.1 Zweck

Dieses Dokument dient als Grundlage zur Beauftragung des berufsbegleitenden Studienganges, Kommunikations- und Medieninformatik des Matrikel 13, mit der Programmierung einer HfTL-APP. Es setzt dabei die Rahmenbedingungen fest.

## 1.2 Hintergründe und Ziele des Projekts

Die Hochschule für Kommunikation Leipzig (HfTL) ist eine private, staatlich anerkannte Fachhochschule. Träger der HfTL ist die HfTL-Trägergesellschaft mbH, eine Beteiligungsgesellschaft der Deutschen Telekom AG. Die Schule befindet sich im Leipziger Stadtteil Connewitz. Es werden sowohl Direkt- als auch duale Studiengänge und berufsbegleitende Studiengänge angeboten. Aufgrund des umständlichen Beschaffens der Noten und Stundenpläne sowie der Termine für Teletutorings, wird eine Smartphone-Applikation benötigt. Diese stellt ein Benutzerinterface für Studenten der HfTL dar, mit dem der Zugriff auf die im QIS hinterlegten Daten vereinfacht.

## 1.3 Produktumfang

Der Betrieb der HfTL-APP muss auf allen gängigen Android-Smartphones ab Version 4.0 möglich sein.
Durch die APP wird den Studenten der HFTL ermöglicht:

- die aktuellsten Nachrichten der HfTL-Webseite lesen
- auf QIS zugreifen
- Studenten sollen ihre(-n) Noten(-spiegel) aufrufen können
- Vorlesungspläne zu lesen
- Raumbelegungspläne abzurufen

Zusätzlich werden folgende Anforderungen gestellt:

- Erweiterbarkeit für weitere Funktionen und Anwender
- spätere IOS-Version

## 1.4 Musskriterien

Zunächst müssen zwingend folgende Punkte des Umfangs erfüllt werden:

- NEWS
- NOTEN
- STUNDENPLAN

## 1.5 Abgrenzungskriterien

Die APP soll später auch für zusätzliche Informationen, wie ein Raumplanung erweiterbar sein. Eine spätere Version für IOS-Geräte ist ebenfalls geplant, ist hier aber nicht Bestandteil.

### 1.5.1 Kostenrahmen

Für die Entwicklung der APP soll auf kostenfreie Opensource-Programme oder auf vordefinierte Klassen der Programmierung zurückgegriffen werden. Außer den personellen Aufwand dürfen keine Zusätzlichen Kosten entstehen.

## 1.6 Definitionen, Akronyme, Abkürzungen

| | |
|---|---|
| **HfTL** | Hochschule für Kommunikation Leipzig |
| **APP** | Kurzform für Applikation |
| **mbH** | mit beschränkter Haftung |
| **QIS** | Qualitätssteigerung der Hochschulverwaltung im Internet durch Selbstbedienung |
| **iCal** | Datenformat zum Austausch von Kalenderinhalten |
| **SoSe15** | Sommersemester 2015 |
| **XML** | Extensible Markup Language |
| **HTTPS** | HyperText Transfer Protocol Secure |
| **AES** | Advanced Encryption Standard |
| **SQL** | Structured Query Language |
| **.apk** | Android application package |
| **MTBF** | mean time between failure |
| **CI/CD** | Corperate Identity/Corperate Design |

## 1.7 Referenzen

- QIS-System: https://qisweb.hispro.de/tel/rds?state=user&type=0
- News der HfTL https://www.hft-leipzig.de/de/studierende/service/news.html

# 2 Allgemeine Übersicht

## 2.1 Beschreibung der Ausgangssituation (Ist-Zustand)

Damit Studenten auf QIS zugreifen kann, müssen diese sich über einen Browser auf der QIS-Seite einloggen und über das unübersichtliche Menü ihre Daten suchen. Studenten haben die Möglichkeit ihre Noten oder einen Notenspiegel einzusehen. Für IPhone-Nutzer gibt es bereits eine kostenpflichtige APP , namens Grades, die Noten und Informationen über die angemeldete Prüfungen auslesen kann. Um die auf QIS hinterlegten Vorlesungspläne abzurufen ist kein Login erforderlich, jedoch ist es notwendig sich umständlich zu dem entsprechenden Studiengang zu navigieren. Einzelne Vorlesungen oder Stundenpläne können als iCal heruntergeladen werden. Ebenso sind die Teletutorien hier zu finden. Die aktuellen Nachrichten der Hochschul-Homepage sind auf einer anderen Seite zu finden, welche öffentlich zugänglich ist.

## 2.2 Produkteinsatz

### 2.2.1 Anwendungsbereiche

Aktuell soll die APP nur für die Studenten der HfTL zugänglich sein, welche ein Android-Smartphone besitzen.

### 2.2.2 Zielgruppen, Qualifikationsniveau

Da bei der Nutzergruppe von Studenten mit Erfahrung im Umgang mit solchen APP's ausgegangen werden kann, wird auch die Oberfläche dementsprechend gestaltet.

## 2.3 Produktfunktionalität

Die APP soll sich mittels regelmäßiger Abfragen der HfTL-Homepage, sowie von QIS, die News, Noten und Stunden- und ggf. Raumbelegungspläne ziehen und die Noten für den Nutzer lokal auf dem Smartphone speichern und entsprechend darstellen. Es soll sichergestellt werden das auf sensible Daten wie z.B. Noten auch nur autorisierte Nutzer Zugang bekommen. Die APP soll in deutscher Sprache dargestellt werden. Evtl. wird sie im Nachgang in andere Sprachen übersetzt.

## 2.4 Randbedingungen

Der zeitliche Rahmen für die Entwicklung und Programmierung dieser APP begrenzt sich auf das Sommersemester 2015 (SoSe15) und dem Studienmodul Software-Engineering. Für die Entwicklung dieser APP, sowie Vertrieb, Programmierung usw. dürfen keine Kosten entstehen. Supportleistungen werden in der Projektphase über das Projektteam geleistet. Die Wartung wird, bis mit der Veröffentlichung die APP der Hochschule kostenfrei zur Verfügung gestellt wird, ebenfalls vom Projektteam übernommen. Die Dokumentation wird ebenfalls vollständig an die Hochschule übergeben. Durch das Projektteam wird es keine

weitere Softwarebetreuung, Wartung oder der gleichen geben. Es finden ebenfalls keine Schulungen oder Einweisungen statt.

### 2.5 Annahmen und Abhängigkeiten

Die APP wird für Android-Geräte ab Version 4.0.3 zur Verfügung stehen. Entsprechend der Vorgaben der Deutschen Telekom AG muss bei der Programmierung der APP, explizit beim Design, auf die Konzernrichtlinien geachtet werden. Es soll zusätzlich auf die Designempfehlungen für Androidgeräte geachtet werden.

### 2.6 Verzögerungen

Durch die strikte Abtrennung des zeitlichen Rahmens auf das SoSe15 darf es über diesen Zeitraum hinaus nicht zu Verzögerungen kommen

## 3 Anwendungsszenarien

### 3.1 Beschreibung aus der Nutzersicht

Die Benutzeroberfläche muss intuitiv bedienbar sein. Der strukturierte Aufbau durch Kategorien (News, Noten, Stundenplan) soll die Übersichtlichkeit erhöhen. Die Logindaten werden verschlüsselt auf dem Smartphone gespeichert und auch verschlüsselt übertragen. Durch eine durchgehende und vollständige Dokumentation soll eine Wartung auch durch spätere Matrikel oder Administratoren der Hochschule möglich sein. Eine Implementierung weiterer Funktionen soll auch im Nachhinein möglich sein.

## 4 Anforderungen

### 4.1 Fachkonzept

Die HfTL-APP wird in Java programmiert um durch Verwendung bestehender Klassen die Erweiterbarkeit und Realisierbarkeit zu vereinfachen. Für das Design werden XML-Stylesheets verwendet. Pull-und Push-Service werden zur Benachrichtigung und Abfrage verwendet.

#### 4.1.1 Überblick über das Gesamtsystem

**Noten**

- die APP soll die Noten lokal auf dem Smartphone nach Semester aufgeschlüsselt anzeigen

- Login über die APP
- Anmeldung über gesicherte, verschlüsselte Übertragung (HTTPS)
- verschlüsselte Speicherung der Daten auf dem Smartphone (AES)
- verschlüsselte Speicherung der Daten auf dem Smartphone (AES)
- Pull-Nachrichten (Einstellbares Intervall und/oder manuell)
- Push-Benachrichtigung
- Nutzer wird mit Hinweismeldung informiert, wenn Noten aktualisiert wurden

Optionale Angaben im Menüpunkt Noten:

- Klassenspiegel
- Notenverteilung
- Anzahl der Teilnehmer
- Notenschnitt
- Anzeige der Creditpoints
- zu erreichende Creditpoints
- erreichte Creditpoints

**Stundenpläne**

- Stundenplan nur für zum Nutzer passenden Studiengang
- Pull-Nachrichten (Einstellbares Intervall und/oder manuelle Abfrage)
- Synchronisierung mit dem Kalender auf dem Smartphone

**News**

- Pull-Nachrichten (manuelle Aktualisierung)
- News von: `https://www.hft-leipzig.de/de/studierende/service/news.html`

#### 4.1.2 Verwendete Bibliotheken von Drittanbietern

- jsoup 1.8.1 – MIT-Lizenz
- iCal4j 1.0.6 – BSD-Lizenz

## 4.2 Anforderungen an die Datenhaltung

Alle gespeicherten Daten müssen vor unbefugtem Zugriff geschützt werden, dafür werden Benutzerdaten in einer SQL-Datenbank gespeichert und durch das Passwort und den Benutzernamen freigegeben. Passwort und Benutzername wurden mit AES verschlüsselt.

### 4.2.1 allgemeine Beschreibung der Daten

Sicherheitsrelevante Daten sind in dem Notenteil der App zu finden. Diese sind der Benutzername, das Passwort, die Prüfungs- und Prüfungsvorleistungsergebnisse, der Klassenspiegel und die Creditpoints. Die öffentlichen Daten sind in den anderen beiden Teilen der App enthalten. Dazu gehören die News mit ihren Terminen und Inhalt und der Stundenplan mit den Veranstaltungen und den dazugehörenden Informationen wie Dozent und Veranstaltungsort.

## 4.3 Anforderungen an die Benutzeroberfläche

### 4.3.1 allgemeine Anforderungen an die Oberfläche

Das Layout richtet sich nach dem Corperate Identity/Corperate Design (CI/CD) der Deutschen Telekom AG, speziell dem der HfTL Trägergesellschaft, stand 14.12.12. Entsprechend sind primär die Farben, sowie Schriftarten vorgegeben. Das Layout zieht sich einheitlich (mit funktionsbedingten Abweichungen) durch alle APP-Teile und soll eine leichte Bedienung begünstigen. Die Größe der einzelnen Elemente (Buttons, Zeilen, Überschriften etc.) ist an den Designempfehlungen seitens Android angelehnt. Auch die Abstände zwischen den einzelnen Elementen wurden – sofern sie mit dem CI/CD in Einklang stehen – entsprechend der Designempfehlung festgelegt. Die Reaktionszeit der Benutzereingaben soll mit möglich wenig Verzögerung verarbeitet und dargestellt werden.

### 4.3.2 Berechtigungen

Die APP wird als Open Source bereitgestellt.
Nicht angemeldete Benutzer können sich die News und den Stundenplan anzeigen lassen.
Angemeldete Benutzer können zusätzlich ihre Noten abrufen.
Die App braucht folgende Berechtigung auf dem Smartphone:

- Netzwerkstatus abrufen: zur Kontrolle ob das Handy online ist
- Internet: APP greift auf das Internet zu
- Telefonstatus: wird zur Verschlüsselung benötigt

### 4.3.3 Bildschirmlayout

siehe Anhang 5.3

#### 4.3.4 Dialogstruktur, Dialogabläufe

Maik/Stefan bitte hier Screenshots der jeweiligen Dialoge einfügen oder in Anhang unter Dialogmasken

### 4.4 Leistungsanforderungen

Es wird von einer maximalen Nutzerzahl von 1000 Studenten ausgegangen. Die Reaktionszeit/Programmstart der APP soll möglichst gering gehalten werden. Das Datenaufkommen soll möglichst gering ausfallen.

### 4.5 Anfordeungen für Inbetriebnahme und Einsatz

#### 4.5.1 Sicherheitsziele

Der Benutzername und das Passwort werden in einen gesicherten Speicherbereich (nur root), mit AES verschlüsselt, gespeichert. Die Datenbank in der die Noten gespeichert werden, liegt ebenfalls in diesem Bereich und darf nur von der HfTL-App geöffnet werden. Die Kommunikation mit der HfTL-Website und QIS erfolgt gesichert per HTTPS.

#### 4.5.2 Installationsprozedur

Der Wirkbetrieb soll nach Möglichkeit über den Android-Playstore realisiert werden. Falls es dabei zu Lizenz- und Kostenproblemen kommt, wird die APP als Android application package (.apk) über die HfTL-Homepage verteilt.

#### 4.5.3 Pilot- bzw. Probebetrieb

Hier müssen die Versionsmanagementprotokolle rein!

#### 4.5.4 Fehlerreaktion, Garantie, Service, »Wiederanlauf«

Die Zuverlässigkeit sollte sich durch eine große mean time between failure (MTBF) darstellen.

#### 4.5.5 Schulungen

Nach Beendigung des Projektes werden keine Schulungen usw. durchgeführt.

### 4.6 Qualitätsanforderungen

#### 4.6.1 Qualitätsmerkmale

Folgende Qualitätsansprüche werden gestellt:

- Hohe Zuverlässigkeit der Software

- schnelle und zuverlässige Verarbeitung der gewünschten Daten
- sichere Datenspeicherung u. -übertragung mittels entsprechender Verschlüsselung
- Fehler werden mit einer entsprechenden Fehlermeldung beantwortet
- Intuitiv benutzbar
- Leicht zu warten und zu erweitern
- Vollständige Dokumentation des Projektes

### 4.6.2 QUALITÄTSSICHERUNG

Zur Qualitätssicherung werden einheitliche Testprotokolle und die entsprechenden Testkriterien erstellt. Es finden regelmäßige Kontrollen durch die Qualitätssicherung statt, um die Einhaltung der gegebenen Standards zu überprüfen. Es findet ebenfalls eine regelmäßige Kontrolle der Dokumentation statt. Die jeweiligen Software-Prototypen werden nach der Prototypisierung zur Fehlererkennung und zum Funktionstest an das Projektteam geschickt. Zur Verbesserung der Software werden die Tests anhand vorgegebener Testkriterien durchgeführt, um standardisierte Testergebnisse zu erhalten. Die Testergebnisse werden zur Auswertung an die entsprechende Projektgruppen zurück gespiegelt.

### 4.6.3 QUALITÄTSNACHWEISE

Während des Projektzeitraumes werden sämtliche Meeting, Tests und Kontrollen protokolliert und zur Dokumentation in einem dafür vorgesehen Bereich abgelegt. Die gesetzten Standards und Vorgaben werden eingehalten. Zum Projektabschluss ist der geplante Umfang erreicht und alle Muss-Kriterien sind enthalten. Die APP ist funktionstüchtig.

#### 4.6.4 Offenlegung der Qualitätskontrollpläne

| Nr. | Anforderung | Verantwortliche |
|---|---|---|
| 1 | Projektrahmen festlegen | gesamtes Team |
| 1.1 | Projektrahmen festlegen | gesamtes Team |
| 1.2 | Architektur | gesamtes Team |
| 1.3 | Vorgehensmodell | gesamtes Team |
| | | |
| 1 | Vollständige Dokumentation | gesamtes Team |
| 1.1 | UML Diagramme erstellen | AD, CM |
| 1.2 | Installationsanleitung und Benutzerhandbuch erstellen | GE, JS |
| 1.3 | Lasten- Pflichtenheft erstellen | SK, SC |
| 1.4 | Protokolle | ML, PK |
| 1.5 | Dokumentation zusammenstellen und gestalten | JS, SC |
| | | |
| 2 | App erstellen | gesamtes Team |
| 2.1 | Design nach HfTL-Vorgaben erstellen | ML, SC |
| 2.2 | Programmierung der APP | GE |
| | | |
| 3 | Qualitätskontrolle | gesamtes Team |
| 3.1 | Testkriterien festlegen | PK |
| 3.2 | Testprotokollelayout erstellen | ML |
| 3.3 | Softwaretests | ML, PK |
| 3.4 | Kontrolle der Dokumentation | PK |
| | | |
| 4 | Anfertigen der Präsentation | gesamte Team |
| 4.1 | Präsentation halten | SC, ML, SK, JS |

#### 4.6.5 Berichte, Protokolle zum Nachweis des Vorgehens gemäss der Qualitätskontrollpläne

Es liegen bei:

- Sitzungsprotokolle etc. (siehe Anlagen)

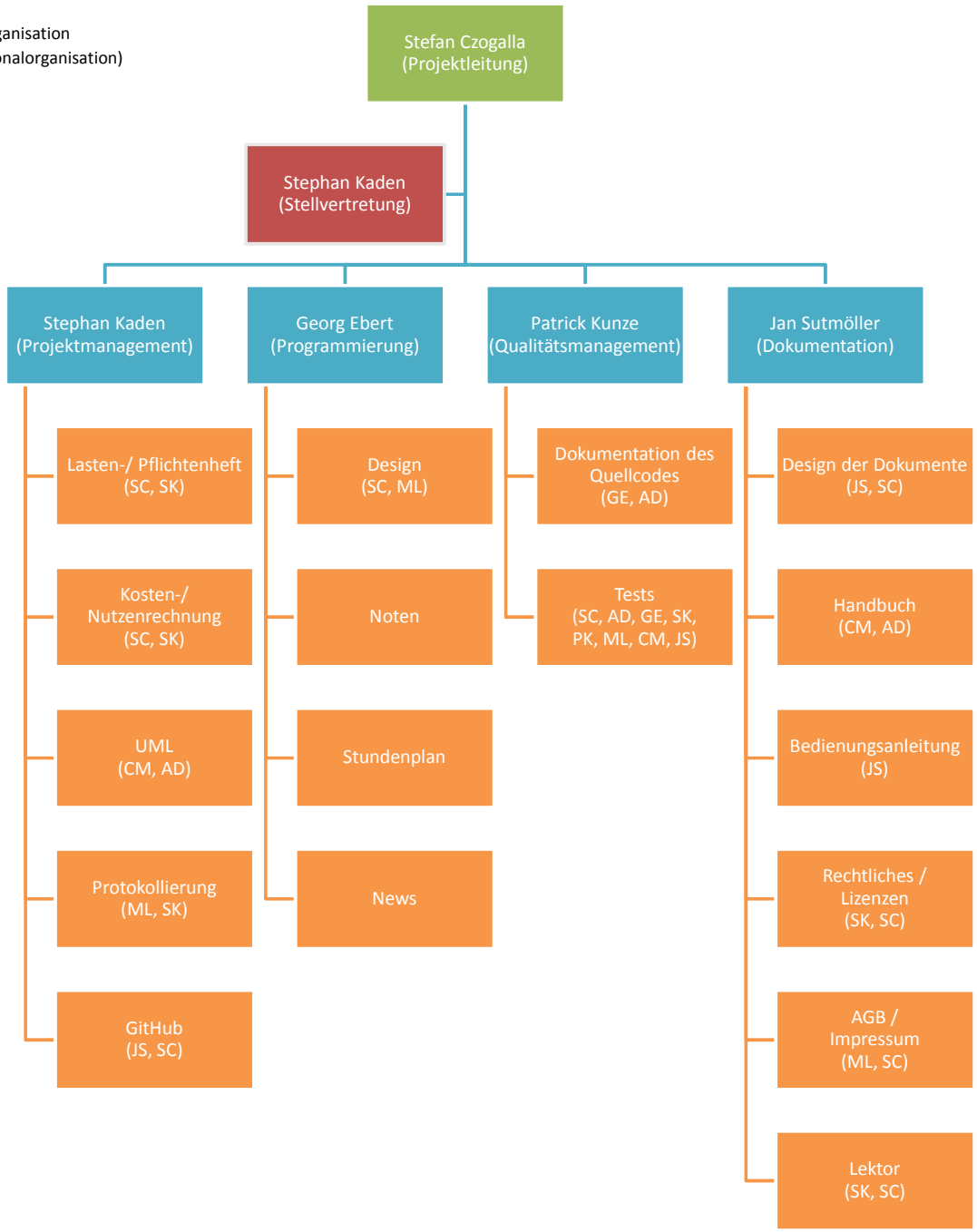
### 4.7 Anforderung an die Entwicklung

#### 4.7.1 Entwicklungs-Umgebung

Für die Entwicklung wird Android Studio inkl. Gradle in der Version 1.x genutzt. Für die Dokumentation und Projektkoordination wird GitHub verwendet. Die Dokumentation wird mittels LaTeX erstellt.

#### 4.7.2 Projekt-Organisation

Projektorganisation
(& Funktionalorganisation)



### 4.7.3 Projekt-Planung

Verweis auf Projektstrukturplan im Anhang

### 4.7.4 Änderungsmanagement

GitHub Plan????

### 4.7.5 Testanforderungen

siehe Anhang 5.4

# 5 Anhang

- Dialogmasken
- Dokumente
- Liste der Softwarelieferungen
- Projektorganigramm
- Projektstrukturplan
- Haupt-Termindaten


# Benutzerhandbuch
# - HfTL-APP -

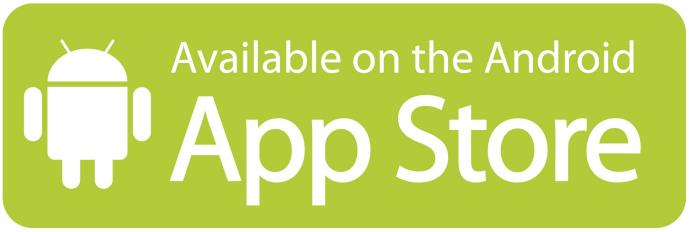
## 5.1 Benutzerhandbuch

### 5.1.1 Funktionsumfang

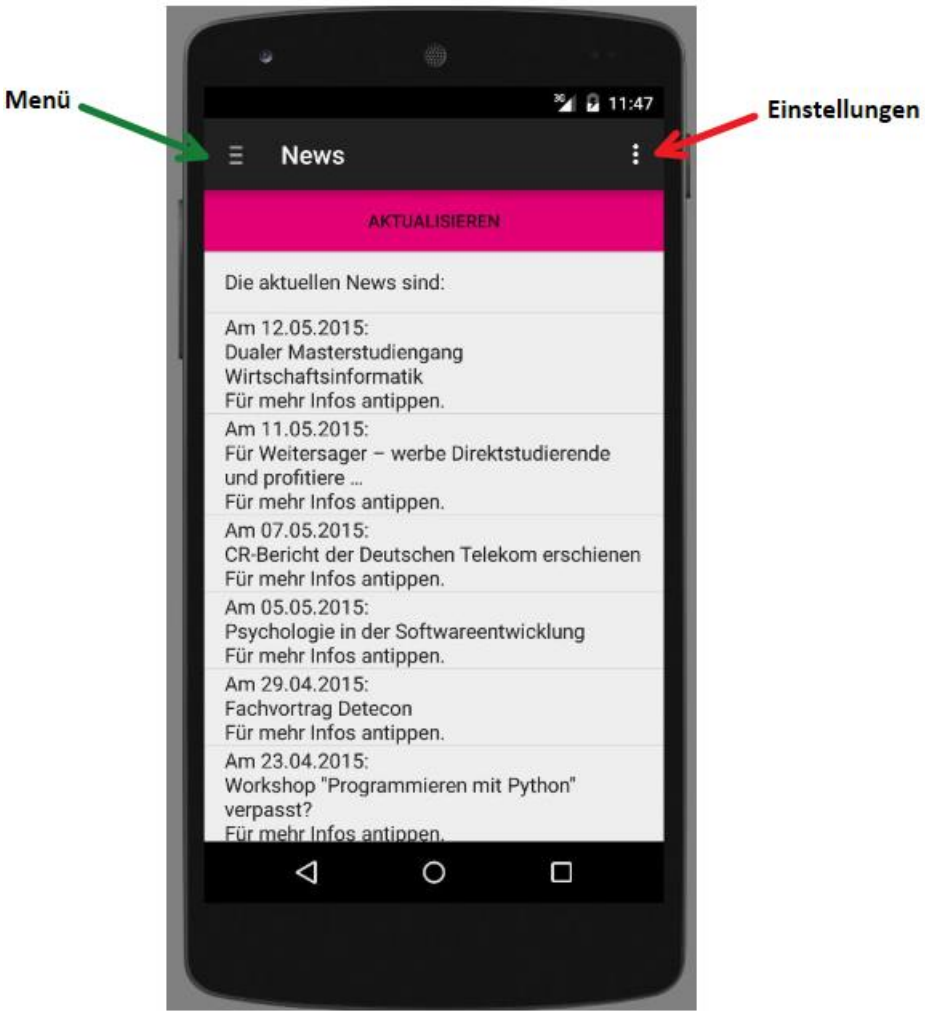
In diesem Dokument wird die Benutzerfunktionen von der HfTL-APP für Android-Geräte beschrieben. Es dient als Benutzerhandbuch für die unterschiedlichen Funktionen der Anwendung und soll Ihnen beim Ausführen von häufigen Aktionen innerhalb der Anwendung Hilfe bieten. Die Installation und Konfiguration von der HfTL-APP wird in einem separaten Dokument behandelt.
Die HfTL-APP ist eine mobile Informationslösung für Android Geräte. Die App kann kostenlos über das Rechenzentrum der Hochschule für Kommunikation-Leipzig bezogen werden.
Die HfTL-APP bietet folgende Funktionen:

- Abfrage der News von der HfTL-Homepage
- Abfrage der Noten aus QIS/HIS nach erfolgreicher Anmeldung an dem betreffenden System
- Abfrage des zu einem Studenten passenden Stundenplans

HINWEIS: In diesem Dokument wird vorausgesetzt, dass die HfTL-APP bereits installiert ist.

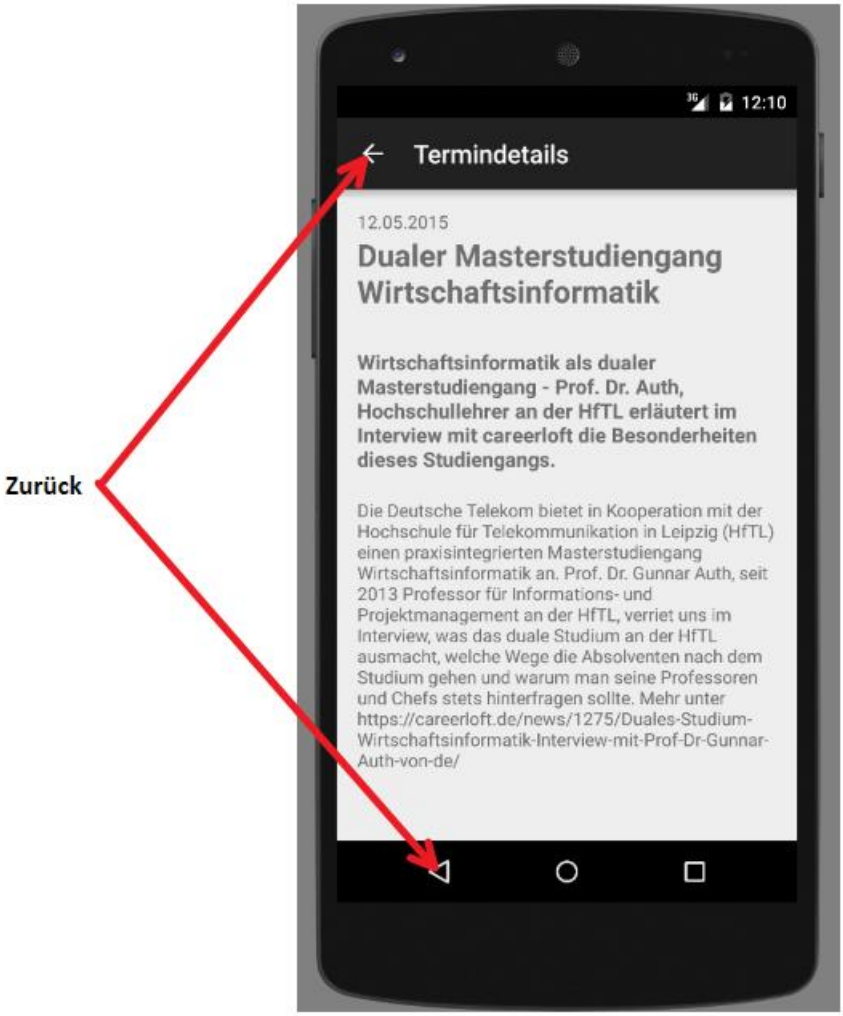


### 5.1.2 STARTBILDSCHIRM




Nach Starten der APP erscheint zunächst die News-Seite. Die News werden bei bestehender Internetverbindung automatisch aktualisiert. Mit klicken auf den Aktualisierungsbutton kann eine manuelle Aktualisierung durch den Nutzer angestoßen werden.
Über den Menü-Button gelangt der Nutzer in das Programm-Menü. Über den Einstellungs-Button gelangt man in das Einstellungs-Menü. Mit Auswählen der einzelnen News gelangt man in deren Detailansicht.

### 5.1.3 NEWSANSICHT



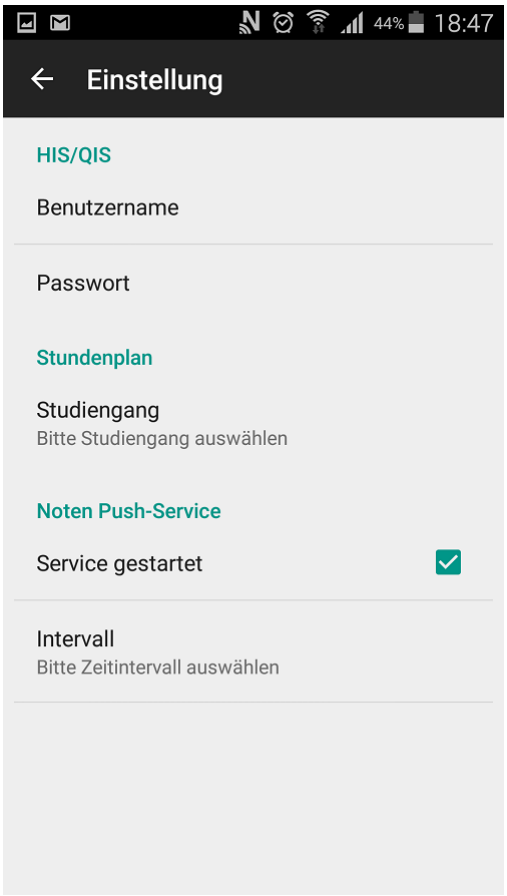

Mit den Zurück-Buttons gelangt man in die vorherige Ansicht.

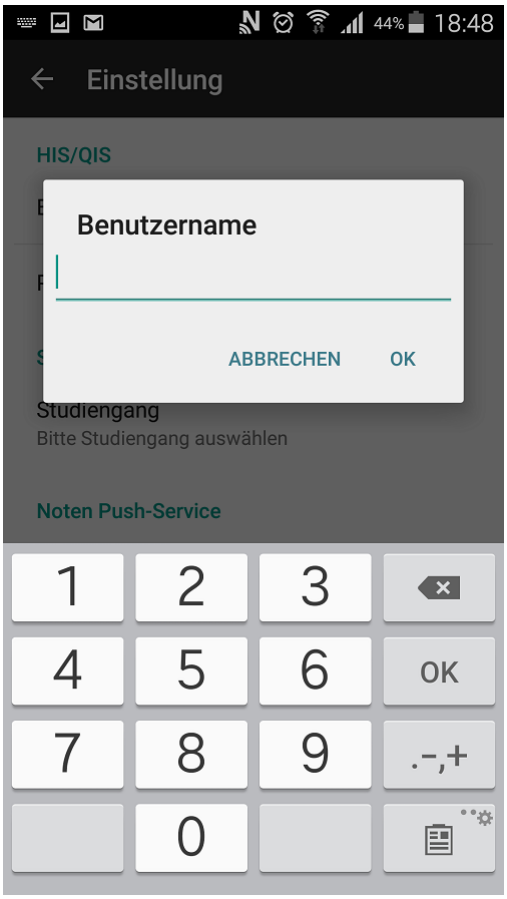
### 5.1.4 Noten

Um die Noten abzurufen geht man zunächst in den Einstellungskontext. Dort kann der Nutzer und das jeweilige Passwort eingegeben werden.

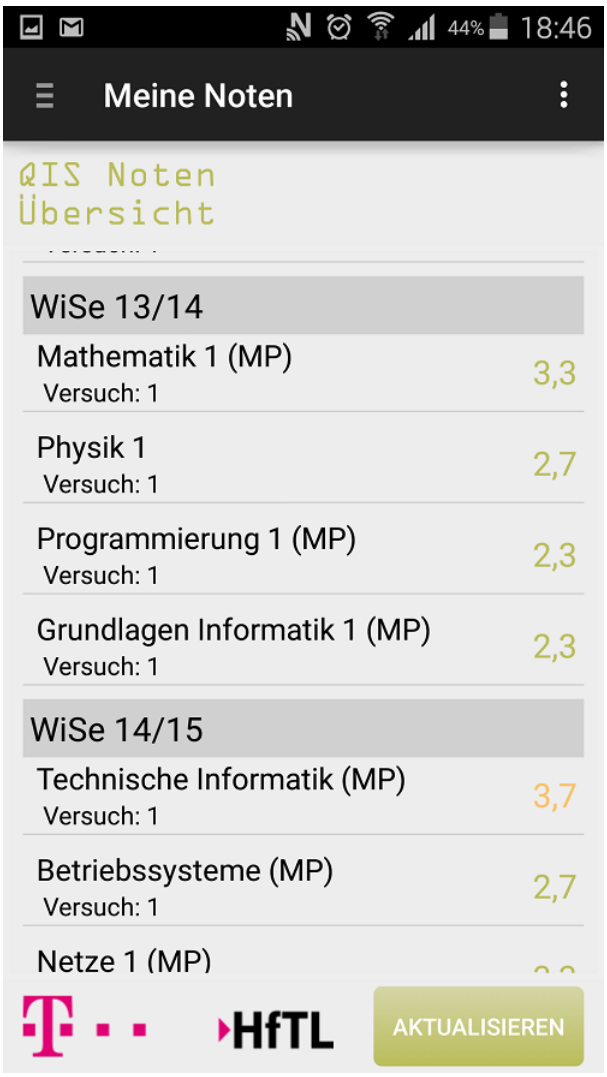
Mit einem Klick auf Benutzername bzw. Passwort öffnet sich ein neuer Kontext welcher zum eingeben des Benutzernamens bzw. Passwortes auffordert.

Die Eingabe wird mit “OK“ gespeichert und mit “Abbrechen“ verworfen. Bei beiden Aktionen schließt sich der Kontext.



Nach erfolgreichem Eintragen des Benutzeraccounts und verlassen der Einstellungen kann über den Menü-Button der Punkt Notenäusgewählt werden. Hier werden die aktuellen Noten aus QIS/HIS geladen und angezeigt. Sollte es beim Anmelden an QIS/HIS zu einem Fehler kommen, erscheint eine Fehlermeldung und die vorher vorgenommenen Einstellungen sollten noch einmal kontrolliert werden.



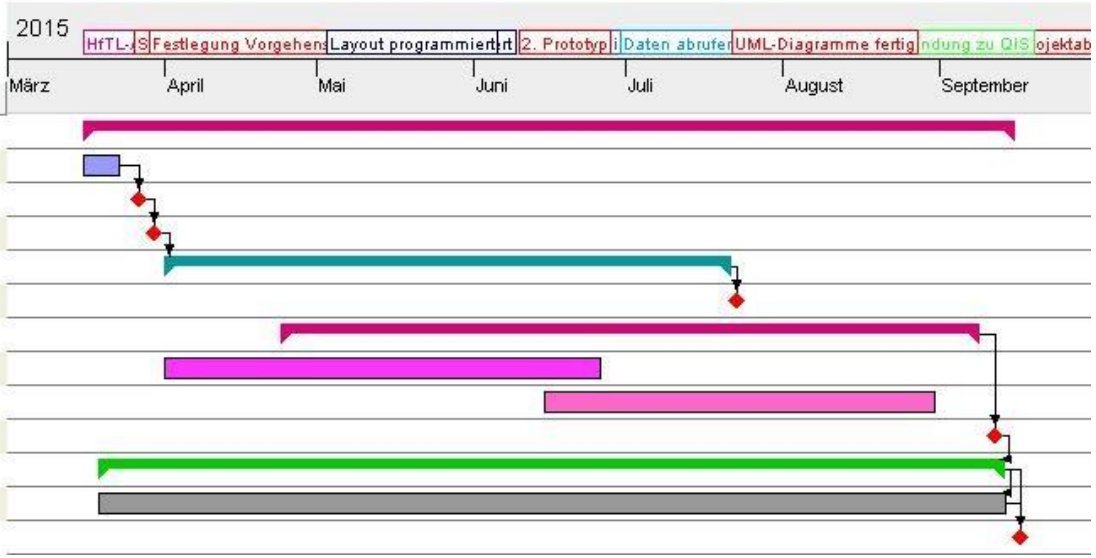
### 5.1.5 Stundenplan

Den Stundenplan erreicht man über das Menü. Hat man in den Einstellungen vorher seinen Studiengang und das Matrikeljahr angegeben, erscheint dazu passende Stundenplan.

## 5.2 GANTT

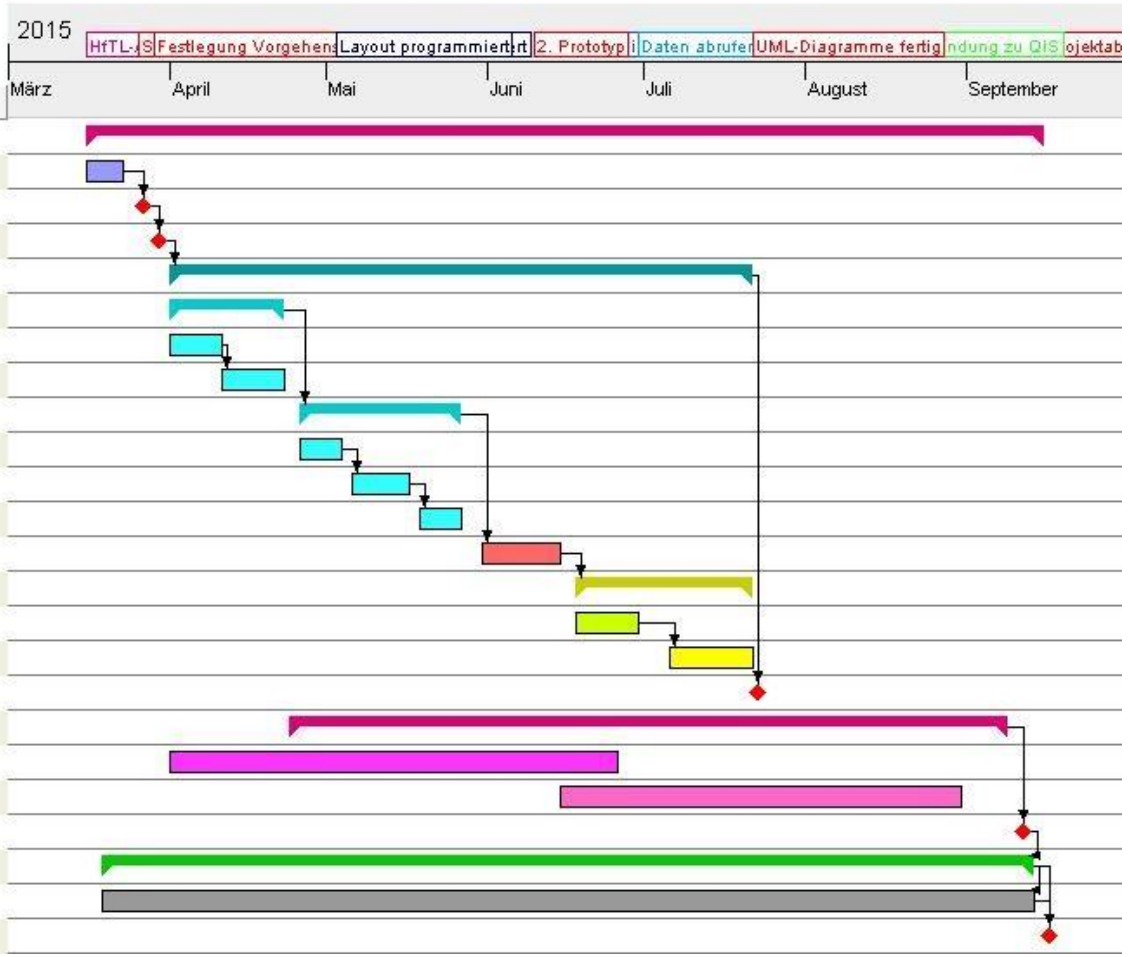
# Allgemeine Übersicht

| Vorgang | Anfang | Ende |
|---|---|---|
| HfTL-APP | 16.03.15 | 15.09.15 |
| SRC | 16.03.15 | 22.03.15 |
| SRC fertig | 26.03.15 | 26.03.15 |
| Festlegung Vorgehensmodell | 29.03.15 | 29.03.15 |
| UML | 01.04.15 | 21.07.15 |
| UML-Diagramme fertig | 22.07.15 | 22.07.15 |
| APP | 24.04.15 | 08.09.15 |
| Festlegung Design/Layout | 01.04.15 | 25.06.15 |
| AGB/Impressum | 15.06.15 | 30.08.15 |
| Finale APP | 11.09.15 | 11.09.15 |
| Dokumentation | 19.03.15 | 13.09.15 |
| Qualitätskontrolle | 19.03.15 | 13.09.15 |
| Projektabschluss | 16.09.15 | 16.09.15 |



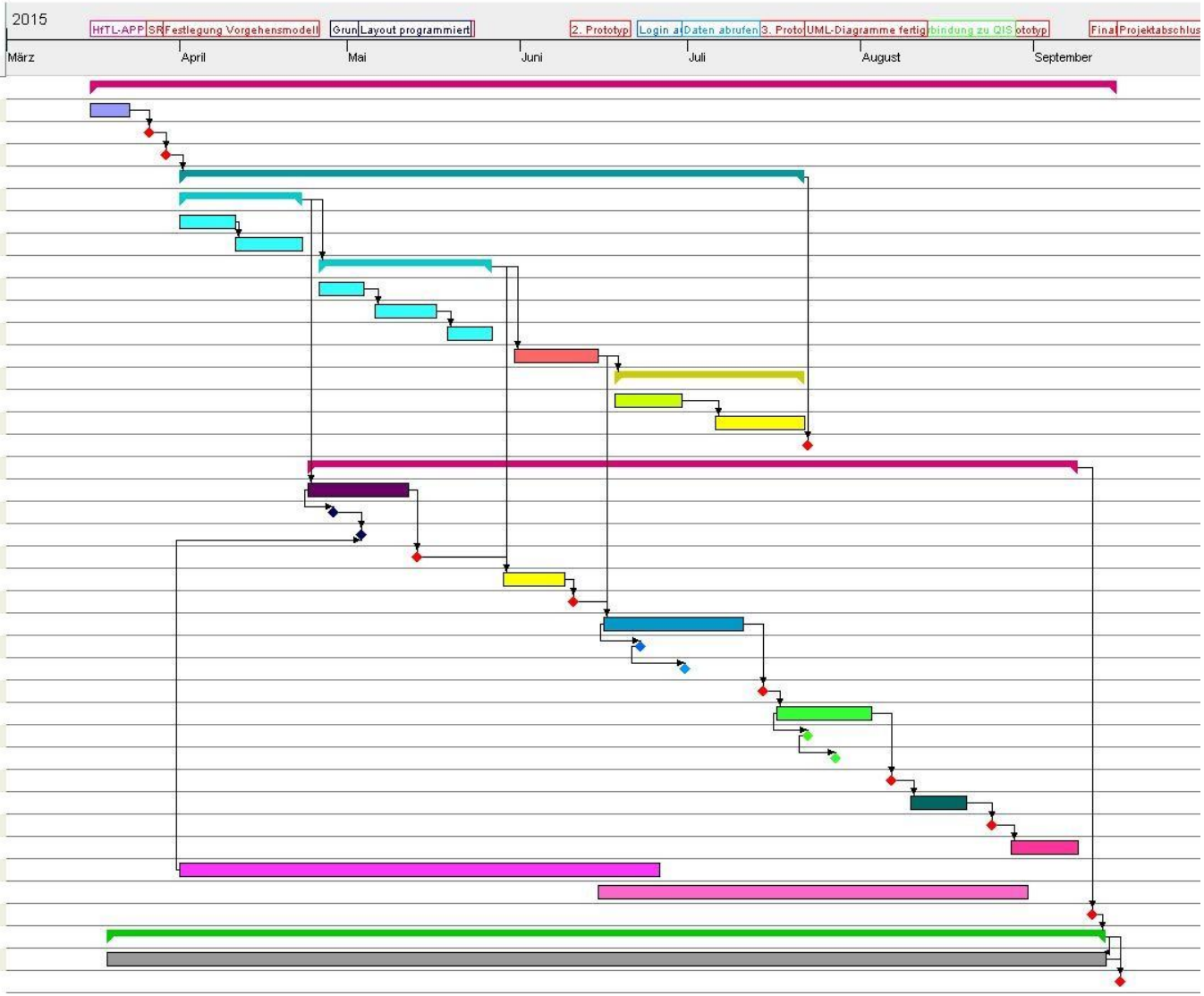
# Detailsicht UML

| Vorgang | Anfang | Ende |
|---|---|---|
| HfTL-APP | 16.03.15 | 15.09.15 |
| SRC | 16.03.15 | 22.03.15 |
| SRC fertig | 26.03.15 | 26.03.15 |
| Festlegung Vorgehensmodell | 29.03.15 | 29.03.15 |
| UML | 01.04.15 | 21.07.15 |
| Klassendiagramm Main | 01.04.15 | 22.04.15 |
| Activity Diagramm starte App | 01.04.15 | 10.04.15 |
| Activity Diagramm Starte App | 11.04.15 | 22.04.15 |
| Klassendiagramm News | 26.04.15 | 26.05.15 |
| Sequenzdiagramm öffne News | 26.04.15 | 03.05.15 |
| Sequenzdiagramm aktualisierung News | 06.05.15 | 16.05.15 |
| Zustandsdiagramm | 19.05.15 | 26.05.15 |
| Klassendiagramm Noten | 31.05.15 | 14.06.15 |
| Use Case Übersicht | 18.06.15 | 21.07.15 |
| Use Case Model | 18.06.15 | 29.06.15 |
| Use Case Starte App | 06.07.15 | 21.07.15 |
| UML-Diagramme fertig | 22.07.15 | 22.07.15 |
| APP | 24.04.15 | 08.09.15 |
| Festlegung Design/Layout | 01.04.15 | 25.06.15 |
| AGB/Impressum | 15.06.15 | 30.08.15 |
| Finale APP | 11.09.15 | 11.09.15 |
| Dokumentation | 19.03.15 | 13.09.15 |
| Qualitätskontrolle | 19.03.15 | 13.09.15 |
| Projektabschluss | 16.09.15 | 16.09.15 |



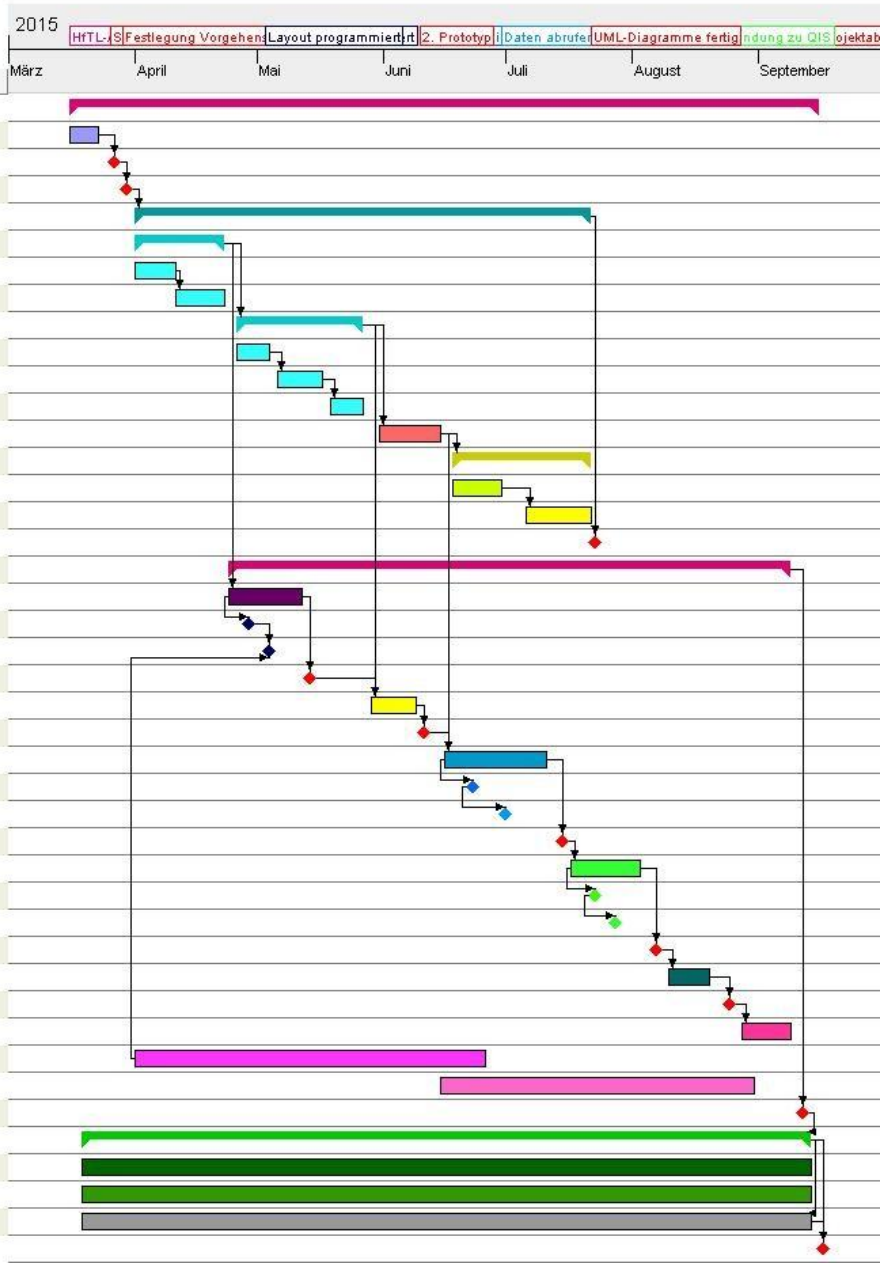
# Detailsicht UML & APP

| Vorgang | Anfang | Ende |
|---|---|---|
| HfTL-APP | 16.03.15 | 15.09.15 |
| SRC | 16.03.15 | 22.03.15 |
| SRC fertig | 26.03.15 | 26.03.15 |
| Festlegung Vorgehensmodell | 29.03.15 | 29.03.15 |
| UML | 01.04.15 | 21.07.15 |
| Klassendiagramm Main | 01.04.15 | 22.04.15 |
| Activity Diagramm starte App | 01.04.15 | 10.04.15 |
| Activity Diagramm Starte App | 11.04.15 | 22.04.15 |
| Klassendiagramm News | 26.04.15 | 26.05.15 |
| Sequenzdiagramm öffne News | 26.04.15 | 03.05.15 |
| Sequenzdiagramm aktualisierung News | 06.05.15 | 16.05.15 |
| Zustandsdiagramm | 19.05.15 | 26.05.15 |
| Klassendiagramm Noten | 31.05.15 | 14.06.15 |
| Use Case Übersicht | 18.06.15 | 21.07.15 |
| Use Case Model | 18.06.15 | 29.06.15 |
| Use Case Starte App | 06.07.15 | 21.07.15 |
| UML-Diagramme fertig | 22.07.15 | 22.07.15 |
| APP | 24.04.15 | 08.09.15 |
| Grundaufbau | 24.04.15 | 11.05.15 |
| Grundgerüst programmiert | 28.04.15 | 28.04.15 |
| Layout programmiert | 03.05.15 | 03.05.15 |
| 1. Prototyp | 13.05.15 | 13.05.15 |
| News programmiert | 29.05.15 | 08.06.15 |
| 2. Prototyp | 10.06.15 | 10.06.15 |
| Noten programmiert | 16.06.15 | 10.07.15 |
| Login auf QIS | 22.06.15 | 22.06.15 |
| Daten abrufen | 30.06.15 | 30.06.15 |
| 3. Prototyp | 14.07.15 | 14.07.15 |
| Sicherung | 17.07.15 | 02.08.15 |
| Verschlüsselte Logindaten | 22.07.15 | 22.07.15 |
| Verschlüsselte Verbindung zu QIS | 27.07.15 | 27.07.15 |
| 4. Prototyp | 06.08.15 | 06.08.15 |
| Stundenplan abrufen | 10.08.15 | 19.08.15 |
| 5. Prototyp | 24.08.15 | 24.08.15 |
| Programmtest | 28.08.15 | 08.09.15 |
| Festlegung Design/Layout | 01.04.15 | 25.06.15 |
| AGB/Impressum | 15.06.15 | 30.08.15 |
| Finale APP | 11.09.15 | 11.09.15 |
| Dokumentation | 19.03.15 | 13.09.15 |
| Qualitätskontrolle | 19.03.15 | 13.09.15 |
| Projektabschluss | 16.09.15 | 16.09.15 |

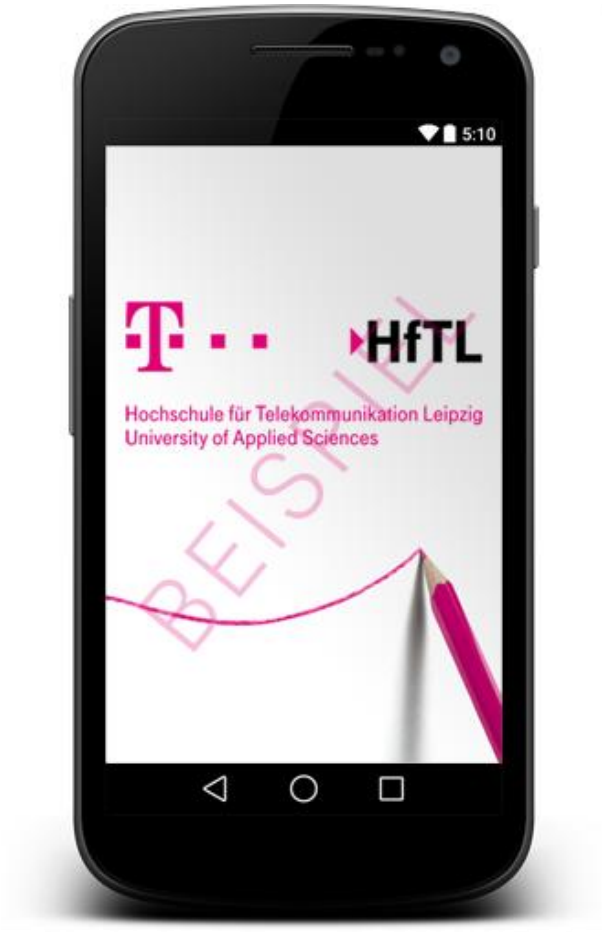


# Gesamtübersicht

| Vorgang | Anfang | Ende |
|---|---|---|
| HfTL-APP | 16.03.15 | 15.09.15 |
| SRC | 16.03.15 | 22.03.15 |
| SRC fertig | 26.03.15 | 26.03.15 |
| Festlegung Vorgehensmodell | 29.03.15 | 29.03.15 |
| UML | 01.04.15 | 21.07.15 |
| Klassendiagramm Main | 01.04.15 | 22.04.15 |
| Activity Diagramm starte App | 01.04.15 | 10.04.15 |
| Activity Diagramm Starte App | 11.04.15 | 22.04.15 |
| Klassendiagramm News | 26.04.15 | 26.05.15 |
| Sequenzdiagramm öffne News | 26.04.15 | 03.05.15 |
| Sequenzdiagramm aktualisierung News | 06.05.15 | 16.05.15 |
| Zustandsdiagramm | 19.05.15 | 26.05.15 |
| Klassendiagramm Noten | 31.05.15 | 14.06.15 |
| Use Case Übersicht | 18.06.15 | 21.07.15 |
| Use Case Model | 18.06.15 | 29.06.15 |
| Use Case Starte App | 06.07.15 | 21.07.15 |
| UML-Diagramme fertig | 22.07.15 | 22.07.15 |
| APP | 24.04.15 | 08.09.15 |
| Grundaufbau | 24.04.15 | 11.05.15 |
| Grundgerüst programmiert | 28.04.15 | 28.04.15 |
| Layout programmiert | 03.05.15 | 03.05.15 |
| 1. Prototyp | 13.05.15 | 13.05.15 |
| News programmiert | 29.05.15 | 08.06.15 |
| 2. Prototyp | 10.06.15 | 10.06.15 |
| Noten programmiert | 16.06.15 | 10.07.15 |
| Login auf QIS | 22.06.15 | 22.06.15 |
| Daten abrufen | 30.06.15 | 30.06.15 |
| 3. Prototyp | 14.07.15 | 14.07.15 |
| Sicherung | 17.07.15 | 02.08.15 |
| Verschlüsselte Logindaten | 22.07.15 | 22.07.15 |
| Verschlüsselte Verbindung zu QIS | 27.07.15 | 27.07.15 |
| 4. Prototyp | 06.08.15 | 06.08.15 |
| Stundenplan abrufen | 10.08.15 | 19.08.15 |
| 5. Prototyp | 24.08.15 | 24.08.15 |
| Programmtest | 28.08.15 | 08.09.15 |
| Festlegung Design/Layout | 01.04.15 | 25.06.15 |
| AGB/Impressum | 15.06.15 | 30.08.15 |
| Finale APP | 11.09.15 | 11.09.15 |
| Dokumentation | 19.03.15 | 13.09.15 |
| Handbuch | 19.03.15 | 13.09.15 |
| Bedienungsanleitung | 19.03.15 | 13.09.15 |
| Qualitätskontrolle | 19.03.15 | 13.09.15 |
| Projektabschluss | 16.09.15 | 16.09.15 |



## 5.3 APP-LAYOUT





| | |
|---|---|
| Seite: | App-Start |
| Hintergrund: | Grafik_Stift.eps |
| Logo: | HfTL_4C_P.eps |
| Auflösung: | 720 x 1280 px |

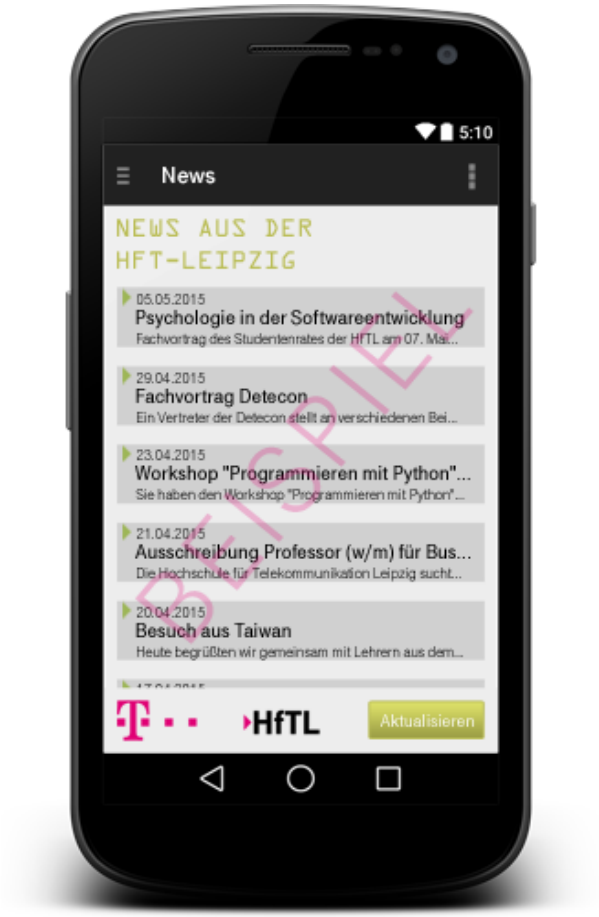
Content: Padding (global): 8dp;
Hintergrundfarbe (global): Grau04;

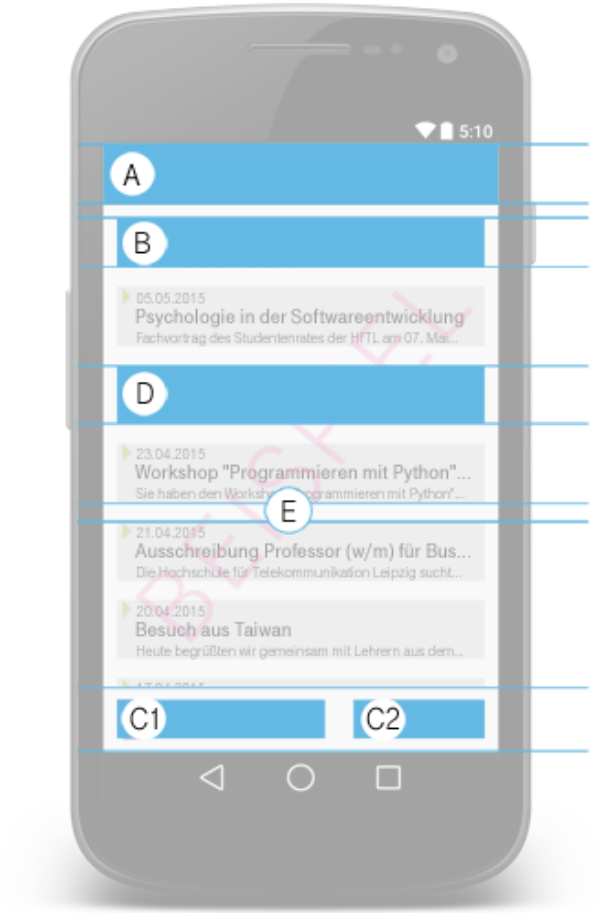
Allgemeine Angaben:

- Die angegebenen Werte beziehen sich auf eine Punktdichte von 320dpi

- sollte die verwendete Schriftart nicht verfügbar sein gilt 'Arial' als Rückfalloption

- Der Seitenabstand von 8dp stellt einen Kompromiss zwischen dem CD der Hochschule und den Vorgaben von Google-Developers dar




**Seite:** News

**A - Header:** Menü-Button-Grafik: ic_drawer.png
Headline-Schrift: wird vom Smartphone gesetzt

**B - Überschrift:** Schrift: OCR A Extended; 24sp; Grün;

**C - Footer**

**C1 - Logo:** HfTL_OB_4C_P.eps

**C2 - Button:** Schrift: Tele-GroteskHal; 18dp; Weiß;
Verlauf: #E4E96E, #9A9C44

**D - Content:** Hintergrund: Grau06;
Grafik: Bulletpoint.png; Größe: 15dp;
Linear Layout Orientation: Vertical;
Padding: 2dp;
Schrift: Datum - Tele-GroteskNor; 15sp;
Überschrift - Tele-GroteskHal; 20sp;
Introtext - Tele-GroteskNor; 15sp;
Margin Left: Überschrift - 15dp;
Introtext - 15dp;

**E - Abstand:** Padding Top + Bottom: 8dp;
Divider: Grau06;



**Seite:** Artikel

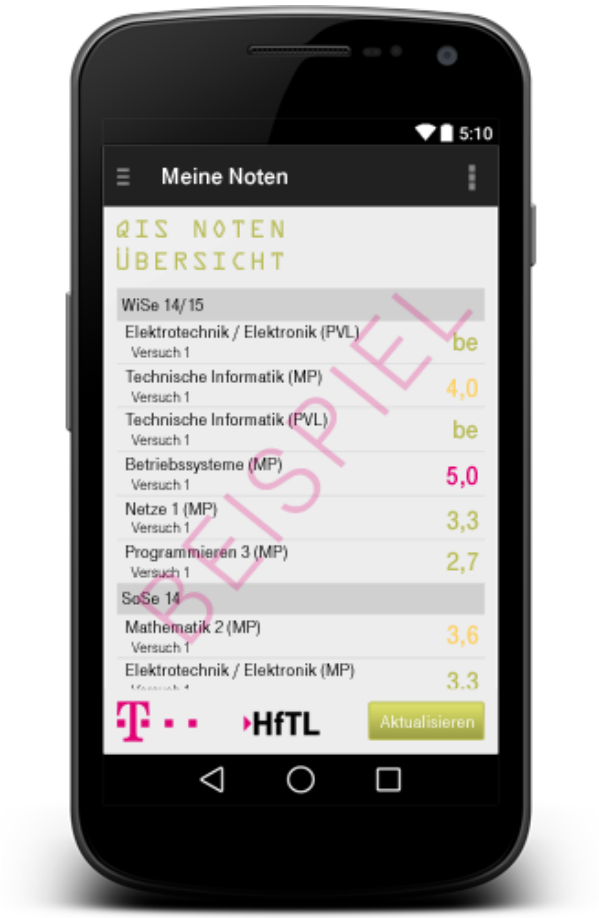
**F - Content:** Linear Layout Orientation: Vertical;

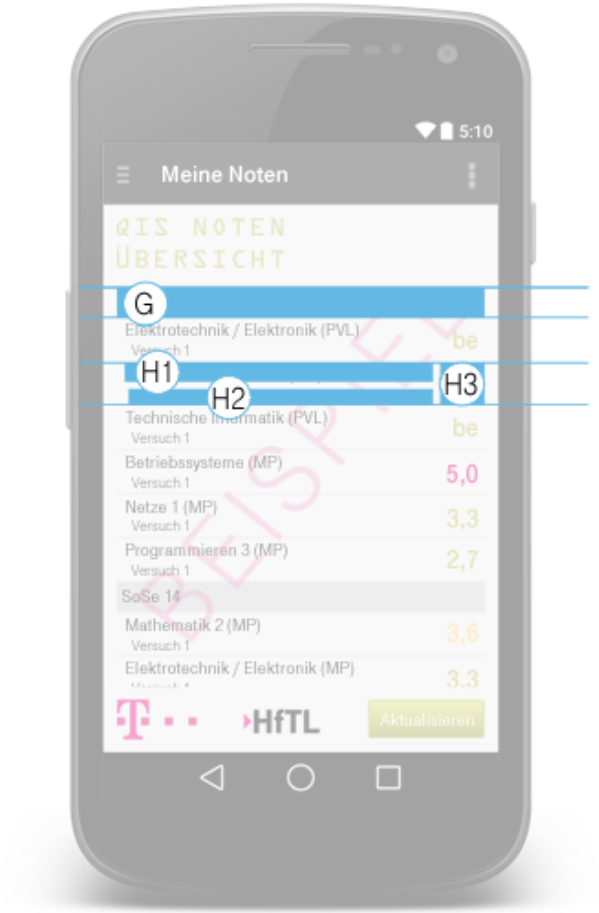
Schrift: Datum - Tele-GroteskNor; 16sp;

Einleser - Tele-GroteskFet; 16sp;

Lesetext - Tele-GroteskNor; 16sp;

URL - Tele-GroteskHal; 16sp;




Seite: Noten (mit Benutzerdaten)

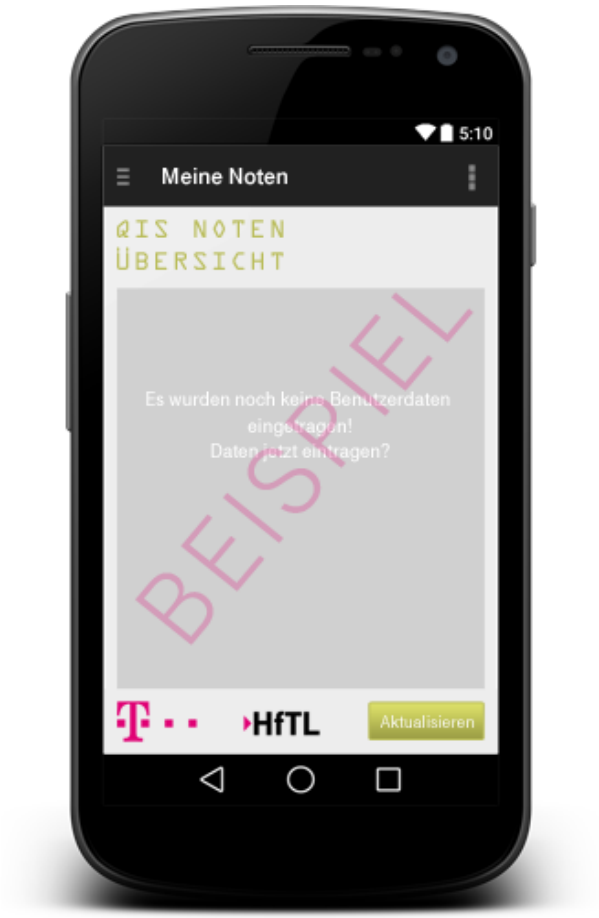
G - Semester: Hintergrund: Grau06;
Schrift: Tele-GroteskHal; 20sp; Schwarz;
Padding: 4dp;

H - Modulnote: Linear Layout Orientation: Vertical;

H1 - Modul: Schrift: Tele-GroteskHal; 18sp; Schwarz;
Margin Left: 8dp;

H2 - Versuchsanzahl:
Schrift: Tele-GroteskNor; 15sp; Schwarz;
Margin Left: 12dp;

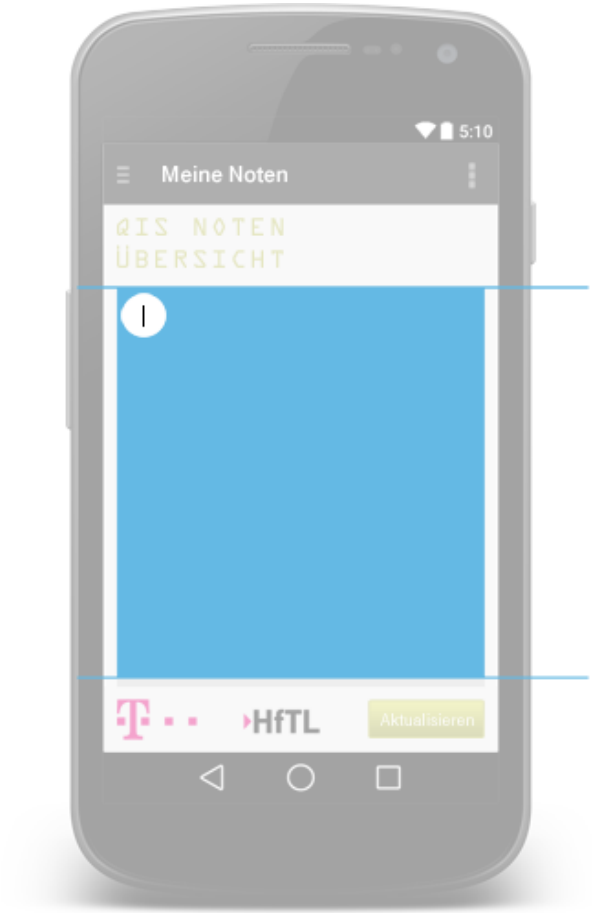
H3 - Note: Schrift: Tele-GroteskHal; 20sp;
Schriftfarbe: Grün - bei Noten <= 3,4
Gelb - bei Noten <= 4,0
Magenta - bei Note 5,0
Layout Gravity: Center;
Margin Left: 4dp;
Padding: 4dp;

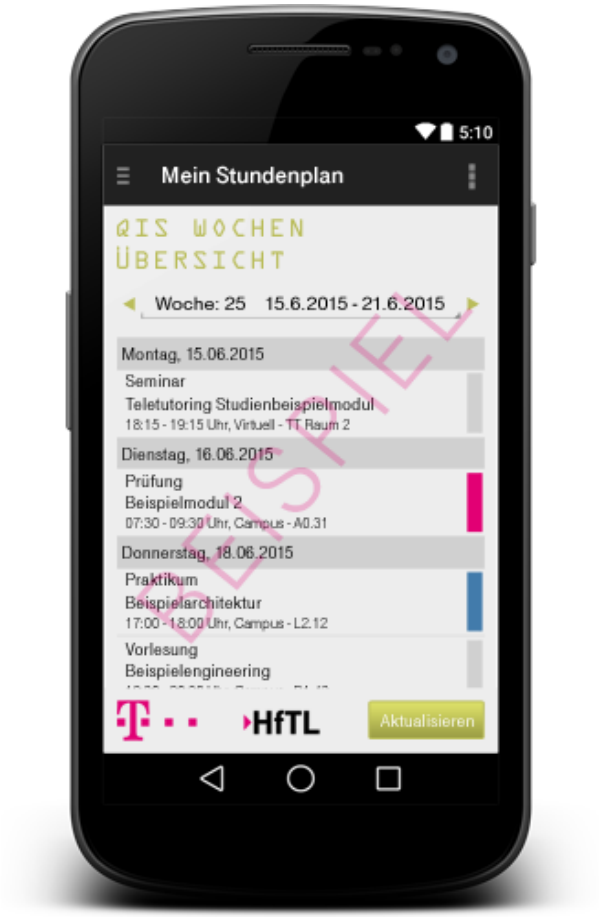



**Seite:** Noten (ohne Benutzerdaten)

**I - Content:** Hintergrund: Grau06;

Schrift: Tele-GrotesktHal; 20sp; Weiß;






Seite: Stundenplan

J - Datumwahl: Linear Layout Orientation: Horizontal;

J1 - Button-Zurück:

Grafik: Bulletpoint_rev.png

Grafik onClick: Bulletpoint_rev_magenta.png

Grafik inaktiv: Bulletpoint_rev_grau.png

Größe: 25dp;

Margin: 8dp;

J2 - Spinner: Schrift: wird vom Smartphone gesetzt

Layout Gravity: Center;

Schriftfarbe: Grün;

J3 - Button-Vor:

Grafik: Bulletpoint.png

Grafik onClick: Bulletpoint_magenta.png

Grafik inaktiv: Bulletpoint_grau.png

Größe: 25dp;

Margin: 8dp;


K - Wochentag: Hintergrund: Grau06;

Schrift: Tele-GroteskHal; 20sp; Schwarz;

Padding: 4dp;

L - Modul: Linear Layout Orientation: Vertical;

L1 - Kategorie:

Schrift: Tele-GroteskHal; 18sp; Schwarz;

Margin Left: 8dp;

L2 - Modul: Schrift: Tele-GroteskHal; 18sp; Schwarz;

Margin Left: 8dp;

L3 - Zeit: Schrift: Tele-GroteskNor; 15sp; Schwarz;

Margin Left: 10dp;

L4 - Kategoriefarbe:

Prüfung - Magenta;

Praktikum - Dukelblau;

Vorlesung, Seminar - Grau06;

Margin Left: 4dp;

Padding: 4dp;

Layout Gravity: Center;

## 5.4 TESTPROTOKOLLENTWURF

# Testprotokoll

| | |
|---|---|
| **Projekt:** | HFTL-APP |
| **Testobjekt:** | *Dateiname/Pfad (inkl. Versionsnummer)* |
| **Testumgebung:** | |
| **Datum:** | |
| **Tester:** | |

| | |
|---|---|
| Testfall 1: | *Kurze Beschreibung was getestet wird* |
| Erwartetes Ergebnis: | *Beschreibung welches Ergebnis erwartet wird* |
| Tatsächliches Ergebnis: | *Beschreibung welches Ergebnis tatsächlich erhalten wurde* |
| Testergebnis: | *Stimmt das tatsächliche Ergebnis mit dem erwarteten Ergebnis überein?* |
| Aufgetretene Probleme: | *Welche Komplikationen sind aufgetreten?* |
| Bewertung: | *Aussage, ob der Test als erfolgreich gewertet werden kann. Wenn nicht, dann muss dies detailliert dargelegt werden. Insbesondere die signifikanten Abweichungen sind hervorzuheben.* |

| | |
|---|---|
| Testfall 2: | *Kurze Beschreibung was getestet wird* |
| Erwartetes Ergebnis: | *Beschreibung welches Ergebnis erwartet wird* |
| Tatsächliches Ergebnis: | *Beschreibung welches Ergebnis tatsächlich erhalten wurde* |
| Testergebnis: | *Stimmt das tatsächliche Ergebnis mit dem erwarteten Ergebnis überein?* |

| | |
|---|---|
| Aufgetretene Probleme: | *Welche Komplikationen sind aufgetreten?* |
| Bewertung: | *Aussage, ob der Test als erfolgreich gewertet werden kann. Wenn nicht, dann muss dies detailliert dargelegt werden. Insbesondere die signifikanten Abweichungen sind hervorzuheben.* |

**Protokollempfänger:** Alle Projektteilnehmer

**Datum:** xx.xx.xxxx

# 6 Entwicklerhandbuch


# Entwicklerhandbuch
# - HfTL-APP -

**Stand: 10. August 2015**

# INHALTSVERZEICHNIS

## 6.1 Allgemeines

Dieses Dokument dient lediglich als Hilfswerkzeug zur (Weiter-)Entwicklung und Wartung der HfTL-App. In diesem Dokument ist der grobe Aufbau, sowie die wichtigsten verwendeten Funktion mitsamt zugehörigen Quelltext erklärt.

Dieses Dokument ist keine Programmieranleitung.

Es empfiehlt sich, gewisse Grundkenntnisse in objektorientierter Programmierung im Allgemeinen und in JAVA, XML und AndroidStudio im Speziellen mitzubringen, um dieses Dokument effizient nutzen zu können.

Standard-Methoden und Klassen sind nicht im Detail erklärt, da das den Rahmen dieses Dokuments sprengen würde. Für tiefer greifende Informationen wird die Android-API empfohlen.

## 6.2 Verwendete Software

**AndroidStudio**

AndroidStudio ist die Standard-Entwicklungsumgebung für Android. Es bietet bereits ein fertiges Gerüst für eine funktionsfähige App an. Das Programm bietet Klassenbibliotheken, Debugger und selbst ein Emulator mit dessen Hilfe Android-Endgeräte auf dem PC virtuell dargestellt werden können.
Das Programm kann kostenlos unter developer.android.com heruntergeladen werden.

**GitHub**

**Gimp**

Gimp ist ein Open-Source Bildbearbeitungsprogramm. Es wurde in dem konkreten Fall genutzt, um Grafiken für die App zu erstellen.
Das Programm kann kostenlos auf der Seite des Herstellers (www.gimp.org) herunter geladen werden.

**Microsoft Office**

## 6.3 Aufbau des Projekts

### 6.3.1 Manifest.XML

Diese Datei ist für jede Android-App zwingend notwendig. Hier werden grundsätzliche Dinge definiert, z.B. Welche Berechtigung diese App benötigt und auf welche Hardware im laufenden Zustand zugegriffen werden muss.
Des Weiteren wird hier auch das package für den Javaquellcode definiert:

Listing 1: AndroidManifest.XML

```
<manifest xmlns:android="..." package="bkmi.de.hftl_app" >
```

Die Zugriffsberechtigungen sind wie folgt definiert:

Listing 2: AndroidManifest.XML

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
    />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Bei der Installation der App wird der Nutzer entsprechend informiert, dass die App Zugriff auf die jeweiligen Funktionen des Endgerätes zugreift.
In der manifest.xml ist ebenfalls eine Übersicht über die verwendeten Verzeichnisse und Komponenten hinterlegt, z.B. für die Activities.
Bei der HfTL-App ist der Verweis für die MainActivity (die Activity, mit der die App startet) für die NewsActivity gesetzt.
Die folgenden Tags wurden bei der App nicht verwendet, sind aber theoretisch für Erweiterungen möglich:

Listing 3: Zugriffsbeispiel

```
<uses-feature android:name="android.hardware.camera" />
```

Das wäre ein Tag, um im laufenden Betrieb auf die Kamera des Telefons zuzugreifen.

### 6.3.2 Ordnerstruktur

#### Database

- beinhaltet die NotenDB.java – Inhalt ist der Connector und die Kernfunktionen um Inhalte der eigentlichen Datenbank zu aktualisieren und zu modifizieren
- NotenTabelle.java – das ist die eigentliche Datenbank, bzw. die eigentliche Definition vom Aufbau der Datenbank

#### Fragmente

- beinhaltet die Fragmente, die von den Activties verwendet werden.

#### Service

- beinhaltet die Datei NotenService.java, die zum Erzeugen von Push-Nachrichten dient, sobald es in der Notenübersicht neue Noten für den jeweiligen Studenten gibt.

#### Help

- beinhaltet diverse Hilfsunktionen, die unter anderem zum Ausführen von Threads dienen. Ressourcen
- unter /src/main/res befindet sich eine Ordnerstruktur, welche XML- und Bilddateien für verschiedenste Anwendungszwecke beinhaltet. Diese werden beim Kompilieren des Projekts in die Ressourcendatei R.java geschrieben. Über diese Datei, wird dann auf die Ressourcen zugegriffen

#### Activities

- Hier werden allgemein Activities und deren Aufbau erläutert...
- ...

#### Fragmente

- Allgemeine Erklärung zu Fragmenten
- ...

## 6.4 Activities

### 6.4.1 NewsActivity.java

Klasse wird mit Fragment extended NavigationDrawerFragment wird benötigt um die Interaktion mit Fragmenten zur Anzeige zu ermöglichen Intent wird benötigt, um andere Activities neben der News Activity zu handlen. mTitle ist der Titel des letztgeladenen Screens

#### onCreate()

- Hier wird das Layout geladen, welches in der zugehörigen xml definiert wurde
- Laden des mNavigationDrawerFragment, zum Abbilden des Menüs am linken Seitenrand
- Laden des Layouts für das Fragment

  Durch das Laden des Layouts für das mNavigationDrawerFragment (durch setUp) wird die Funktion onNavigationDrawerItemSelected gerufen, weil in der Layoutdefintion Item 1 der Listview ausgewählt wird.

#### onNavigationDrawerItemSelected()

- Durch das Aufrufen der Funktion wird das eigentliche Fragment für den FragmentManager ausgewählt (durch die switch case Anweisung) und abschließend durch das commit aktiv geladen.

#### onSectionAttached()

- Je nach Auswahl wird hier mTitle aktualisiert.

#### restoreActionBar()

- Dient zum Aktualisieren der Titelleiste (durch setTitle und mTitle als Argument)

#### onCreateOptionsMenu()

- Erstellt das Menü oben rechts (drei Punkte) und befüllt es mit den Daten aus der /res/menu/news.xml

#### onOptionsItemSelected()

- Überprüft welches Element aus dem Menü ausgewählt wurde

### 6.4.2 EINSTELLUNGSACTIVITY.JAVA

#### onCreate()

- Einbinden der "einstellung.xml" (beinhaltet Definitionen für Stringvariablen, Listen und Checkboxen)
- Setzen von Startwerten für shared, check und list
- Ausführen von registerPreferenceListener()

#### registerPreferenceListener()

- Es wird ein anonymer Listener erstellt, der auf Änderungen in der SharedPreference.xml reagiert
- In der Methode onSharedPreferenceChanged(SharedPreferences prefs, String key) werden die Änderungen abgefangen
- Listener wird am SharedPreferences Objekt registriert

#### testeBenutzerdaten()

- Funktion zum Überprüfen von Anmeldedaten
- via TextSecure ts wird Ver- und Entschlüsselung gewährleistet

#### keineBenutzerdaten()

- Funktion zum Ausgeben, dass die Anmeldeinformationen falsch eingegeben wurden

## 6.5 NewsClickedActivity.java

### 6.5.1 Allgemein

Diese Activity wird geladen, sobald in der News-Übersicht (NewsActivity.java) ein Eintrag geöffnet wird.
Der Inhalt wird durch NewsResolver und dessen Funktion getDetailsStringArray in das String-Array s geladen.

#### onCreate()

- Laden des in der zugehörigen xml definierten Layouts (activity_ news_ clicked.xml)
- Einbinden der Ëxtras"(Übergebene Variablen)
- Zuweisung der URL aus dem rufenden NewsFragment
- Zuweisung der TextViews durch R.java
  - Einbinden der verwendeten Schriftarten durch

```
Typeface.java();
```

  - Zuweisung der Schriftarten an den jeweiligen TextView durch

```
setTypeface();
```

  - Aufruf des DetailHelpers durch

```
new Detailhelper().execute();
```

### 6.5.2 Klasse DetailHelper

#### onPreExecute()

- Anzeige eines ProgressDialogs, um den Anwender zu informieren

#### onPostExecute()

- Ressourcen des ProcessDialogs werden freigegeben
- Beschriftungen der Listboxen werden gesetzt.

#### doInBackground()

- Neuinstanzierung eines NewsResolvers mit übergebener Url, um durch die Funktion getDetailsStringArray das Array s mit Daten zufüllen

```
s[0]=elements.get(0).child(1).text()+"\n";    //Ueberschrift
s[1]=elements.get(0).child(2).text()+"\n";    //Subhead
s[2]=elements.get(0).child(0).text();         //Zeit
s[3]=elements.get(0).child(3).text();         //Text
```

## 6.6 FRAGMENTE

### 6.6.1 NEWSFRAGMENT

Initialisierung erfolgt durch „NewInstance“, die aus der NewsActivity heraus gerufen wird. (siehe Funktionsaufruf -> onNavigationDrawerItemSelected) Durch das .commit wird diese neue Instanz der Klasse dann geladen.

**Überschriebene Funktionen:**

#### OnAttach()

- Verknüpfung des NewsFragments mit der MainActivity

#### OnCreateView()

- Hier wird lediglich das Layout der View des Fragments geladen und angewendet

#### onActivityCreated()

- es wird geprüft, ob es SavedInstances (bereitsgeladene Inhalte) gibt
- wenn JA:
  - die Informationen werden aus dem ARRAYSPEICHER geholt und angewandt
- wenn NEIN:
  - die Funktion „zeigeNews“ wird gerufen, welche die aktuellsten News vom Server lädt

#### onStart()

- in dieser Funktion wird nur noch der Listener für den Aktualisierbutton mit der Schaltfläche verknüpft. Als onClick-Event wird dann lediglich die Funktion „zeigeNews“ gerufen.

**standAllone-Funktionen**

#### isOnline()

- Diese Methode prüft durch einen Connectivity Manager, ob eine Verbindung zum Internet besteht

#### zeigeNews()

- Zunächst wird durch „isOnline“ geprüft, ob eine aktive Netzverbindung besteht

- wenn NEIN:
  - Es wird ein Hinweis an den Nutzer ausgegeben
- wenn JA:
  - Es wird eine Instanz der Klasse NewsHelper erstellt, welche dann als Hintergrund-Task ausgeführt wird, um ein Einfrieren der App zu verhindern.

### 6.6.2 Klasse NewsHelper

- Klasse mit asynchroner Task-Ausführung
- Nach dem Aufruf der Methode *zeigeNews* wird durch den Unterfunktionsaufruf *.execute* zunächst die Funktion *doInBackground* ausgeführt, wo eine neue Instanz des NewsResolvers erstellt wird.

```
private void zeigeNews(){
 NewsHelper nh = new NewsHelper();
 nh.execute();
}
```

- Durch *getTermineStringArray* des NewsResolvers wird ein String-Array zurückgegeben.
- Danach wird die Methode *onPostExecute* gerufen.

#### onListItemClick()

- Es wird ein intent verwendet um die *NewsClickedActivity* zu starten, als "Übergabeparameter" wird *putExtra* verwendet, im Falle der App die URL zu dem angeklickten Event.

```
 intent = new Intent(getActivity(), NewsClickedActivity.class)
    ;
 intent.puExtra(TERMINDETAIL, newsResolver.getURLAsString(
    position));
 startActivity(intent);
```

#### onPostExecute()

- Überprüfung ob das Fragment noch aktiv ist

```
if(getActivity()==null) return;
```

- einbinden des *CustomAdapterNews* in die ListView

```
setListAdapter(new CustomAdapterNews(getActivity(), ...);
```

### 6.6.3 NavigationDrawerFragment

Dieses Fragment bildet das Menü auf dem linken Rand der App ab und aktiviert den Button mit dem man in die Einstellungen gelangt.

#### onCreate()

In dieser Methode werden die Einstellungen der Activity übernommen und der Drawer ausgewählt. Außerdem wird geprüft ob eine gesicherte Instanz vorhanden ist, aus der dann das zuletzt ausgewählte Fragment ermittelt wird und über die Funktion *selectItem()* aufgerufen wird. Falls keine gespeicherte Instanz existiert, wird die "0" (*NewsFragment*) als Standardwert übergeben.

#### onActivityCreated()

Hier wird das Menü für das aktuelle Fragment aktiviert indem an *setHasOptionsMenu()* "true" übergeben wird.

#### onCreateView()

In dieser Funktion wird das Design für die Actionbar (als Listview) und die dazugehörigen Menüpunkte festgelegt. Zudem wird hier der ClickListener initialisiert, der dann den ausgewählten Eintrag an *selectItem()* übergibt.

#### isDrawerOpen()

Diese Methode prüft ob der Drawer bereits offen ist und liefert einen Wahrheitswert zurück.

#### setUp()

Es werden hier folgende Einstellungen vorgenommen:

- Einstellungen zum Design
- Aktivierung des HomeButtons und dessen Animation beim Draufklicken
- Zusammenführung der *ActionBar* und des *NavigationDrawers*
- weitere Einstellungen zum Drawer

Dann wird die Konfiguration in *mDrawerToggle* abgespeichert

#### selectItem()

Hier wird die Animation auf den angeklickten Menüpunkt ausgeführt. Zudem wird der Drawer geschlossen und die Position des angeklickten Punktes an mCallbacks.onNavigationDrawerIte
übergeben.

#### onAttach()

Diese Methode setzt setzt den Zeiger mCallbacks auf die Activity.

#### onDetach()

Hier wird der Zeiger *mCallbacks* auf "null" gesetzt.

#### onSaveInstanceState()

Diese Funktion sichert die aktuelle Instanz.

#### onConfigurationChanged()

Bei einer Änderung in den Einstellungen konfiguriert diese Methode *mDrawerToggle* um.

#### onCreateOptionsMenu()

Diese Funktion legt das Design, aus einer XML-Datei für das Menü Einstellungen, fest.

#### onOptionsItemSelected()

Hier wird der Button, über den man zu den Einstellungen gelangt, aktiviert und mit dessen Klasse verknüpft.

#### NavigationDrawerCallbacks()

Hier wird die ausgewählte Menüpunkt-ID an die Activity übergeben.

### 6.6.4 NOTENFRAGMENT

#### onCreateView()

- Laden des entsprechenden XML-Layouts *fragment_ noten.xml*
- Laden und Zuweisung der Schriftart des TextViews für die Überschrift des Fragments

#### onAttach()

- Aufruf der Datenbank, in der die Noteneinträge abgelegt werden

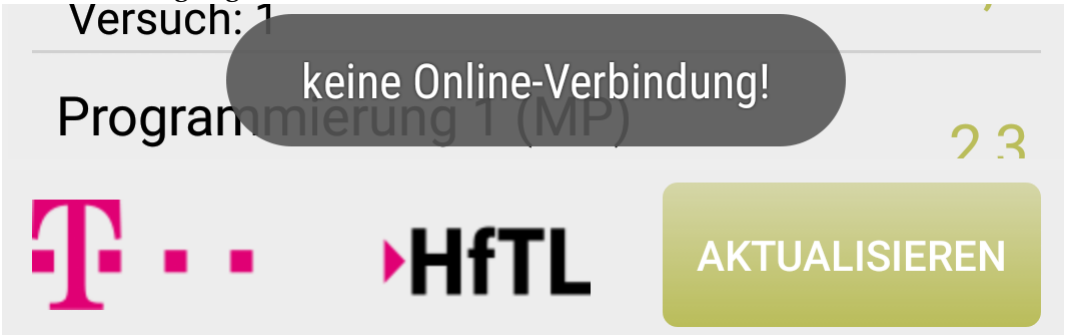
#### onDetach()

- Schließen der Datenbank

#### onStart()

Erstellen des Buttons zum Aktualisieren Mittels *OnClickListener* für den Button, wird über die Methode *testeBenutzerdaten()* überprüft, ob die Benutzerdaten für QiS eingetragen wurden. Andernfalls erfolgt die Ausgabe mittels der Methode *keineBenutzerdaten()*, dass diese nicht eingetragen wurden. Sofern die Benutzerdaten eingetragen wurden, und eine Onlineverbindung zu QiS besteht, werden die Noten erneut

abgerufen. Ist keine Verbindung zu QiS vorhanden, wird der Nutzer mittels Toast-Benachrichtigung darüber informiert:




### getNoten()

Liest in der Notendatenbank alle Werte in der Spalte NOTENABFRAGE aus und schreibt diese in das String-Array s, welches auch übergeben wird.

### getSemester()

Liest in der Notendatenbank alle Werte in der Spalte SEMESTERABFRAGE aus und schreibt diese in ein String-Arrays, welches auch übergeben wird. Vor dem Schreiben des Arrays wird noch auf doppelte Werte geprüft. Sobald ein Wert in der Spalte doppelt ist, bleibt der Eintrag an der entsprechenden Position des String-Arrays NULL. (Damit wird eine Auflistung der Noten und Fächer, aufgeschlüsselt nach Semester realisiert, siehe *CustomAdapterNoten.java*.)

### getFach()

Liest in der Notendatenbank alle Werte in der Spalte FACHABFRAGE aus und schreibt diese in das String-Array s, welches auch übergeben wird.

### getVersuche()

Liest in der Notendatenbank alle Werte in der Spalte VERSUCHABFRAGE aus und schreibt diese in das String-Array s, welches auch übergeben wird.

### Hinweis:

*Die String-Arrays sind nur lokal in den jeweiligen Methoden definiert, weshalb diese - der Einfachheit halber - alle den Namen "s" erhielten. Tatsächlich werden hier vier unterschiedliche String-Arrays befüllt.*

### setzeListview()

Hier werden zunächst entsprechende String-Arrays durch die jeweiligen Methoden zur Abfrage in der Datenbank befüllt:

```
notenList = getNoten();
semesterList = getSemester();
fachList = getFach();
versuchList = getVersuche();
```

Anschließend werden diese String-Arrays an den *CustomAdapterNoten.java* übergeben. Damit wird eine individuelle Befüllung und Formatierung der Liste mit den ausgelesenen Werten aus der Datenbank realisiert.

#### keineBenutzerdaten()

– NotenHelper (Class)

### 6.6.5 StundenplanFragment

#### newInstance()

Erstellt ein StundenplanFragment und "steckt" die aktive Position (aus dem Navigation Drawer) in das "Bundle args" welches als Argument im Fragment übergeben wird.

#### onCreateView()

Laden des entsprechenden XML-Layouts fragment_ noten.xml Laden und Zuweisung der Schriftart für das TextView für die Überschrift des Fragments

#### onViewCreated()

falls im "Stundenplanspeicher" Daten vorhanden sind werden diese geladen Methode für das Dropdownmenü wird gerufen und Array für "events" erstellt anonyme Listener für die Buttons werden erstellt

#### erzeugeDropdown()

Dropdown aus xml einen Objekt zuweisen Listener für Dropdown (als anonymer Listener) erzeugen und registieren beim registieren wird die Methode onItemSelected aufgerufen und ein StundenplanHelper ausgeführt Mittels eines "Calendar", "Date" und zwei "SimpleDateFormat" wird das Dropdownmenü befüllt, indem die Daten in eine String-List eingefügt werden (list.add(temp)) Dropdown mit Liste verknüpfen

#### keinStudiengang()

Prüft ob in den Einstellungen der Studiengang eingetragen wurde falls nein –>Fehlermeldung

#### erstelleStundenplan()

Erzeugt die Ausgabe des Stundenplans und fügt sie in den ListView ein Falls keine Daten vorhanden sind wird "keine Daten" ausgegeben

– StundenplanHelper (class)

#### onPreExecute()

erzeugt ein Ladebalken

#### onPostExecute

falls das Fragment noch aktiv ist wird die Methode erstelleStundenplan() gerufen Ladebalken wird entfernt

### doInBackground

erzeugt ein StundenplanResolver befüllt das “StundenplanEvents-Array Events“ mit Daten, durch die Methode “erzeugeStundenplan(String woche)” des StundenplanResolvers falls ein Fehler auftritt wird ein Event erstellt in dem keine Daten sind

## 6.7 CustomAdapter

### 6.7.1 Allgemein

Die CustomAdapter kommen da zum Einsatz, wo eine ListView genutzt und individuell befüllt werden muss.
Alle CustomAdapter entstehen durch Vererbung aus der Klasse BaseAdapter.java (s. AndroidAPI). Entsprechend sind einige Methoden vorgegeben, die initialisiert werden müssen (mit Rückgabewert), jedoch für die HfTL-App nicht (zwingend) von Bedeutung sind:

```
public int getCount()
public Object getItem()
public long getItemID()
```

Darüberhinaus ist der weitere Aufbau bei den drei verwendeten CustomAdaptern gleich. (Auf die jeweiligen Besonderheiten wird dann konkret im Abschnitt der jeweiligen CustomAdapter eingegangen):

Anfangs werden neue String-Arrays initialisiert, die mit den übergebenen Werten aus den Activities befüllt werden sollen.

```
String [] name;
```

Die abstrakte Klasse Context wird initialisiert und deklariert. (s. AndroidAPI)

```
Context context;
```

Anschließend wird der Konstruktor der Klasse mit den jeweiligen übergebenen Parametern aufgerufen.
Welche Parameter genommen werden, hängt vom jeweiligen Aufruf des CustomAdapters in der entsprechenden Activity ab. Nun werden die übergebenen Parameter den korrespondierenden Variablen zugeordnet.

```
public CustomAdapter (Activity nameActivity, String[] nameList)
    {
 name = nameList;
 context = nameActivity;
}
```

#### class Holder

Dies ist ein Container mit TextViews

```
TextView tv_name;
```

Dieser Container wird dann als jeweilige Zeile in der ListView der zugehörigen Activity dargestellt.

**public View getView()**

erstellt einen neuen Container

```
Holder holder = new Holder;
```

erstellt eine View rowView

```
View rowView;
```

Das Layout der View und damit jeder Zeile in der ListView wird durch das Laden der entsprechenden XML namelist formatiert:

```
rowView = inflater.inflate(R.layout.name_list, null);
```

Die TextViews des Containers werden den entsprechenden TextViews in der XML zugeordnet:

```
holder.tv_name=(TextView)
rowView.findViewbyId(R.id.namelist_name.xml);
```

Das TextView wird mit dem entsprechenden Wert des zugehörigen String-Arrays befüllt:

```
holder.tv_name.setText(name[position]);
```

Der nun befüllte Container wird nun als rowView übergeben.

```
return rowView;
```

### 6.7.2 CustomAdapterNews.java

Dies ist der simpelste CustomAdapter der HfTL-App. Er wird verwendet um die News-Liste (also die Übersicht der News aus der HfTL-Homepage) zu formatieren.
Dabei werden drei String-Arrays (date, headline, content) in das zugehörige TextView der rowView übergeben. Der Inhalt dieser Arrays wird mittels der Klasse Newshelper, die wiederum die Methoden der Klasse NewsResolver aufruft, befüllt.

### 6.7.3 CustomAdapterNoten.java

Dieser CustomAdapter wird für die Befüllung der ListView des Notenfragments benutzt.
Hier werden vier String-Arrays (subject, trys, mark, semester) mit den Daten aus der Notendatenbank befüllt.
Entsprechend des Inhalts, werden die zugehörigen TextViews noch gesondert formatiert.
Ist der Inhalt an der Position des String-Arrays “NULL“, wird das zugehörige TextView auf “GONE“ gesetzt. So wird realisiert, dass mehrere Fächer unter dem selben Semester gelistet sind, ohne dass ein leeres (dunkelgraues) TextView erscheint.

```
if(semester[position]==null){
 holder.tv_semester.setVisibility(TextView.GONE);
}
```

Auch für die jeweiligen Noten gibt es eine gesonderte Formatierung:

- Note schlechter als 5.0: magenta
- Note schlechter als 3,4: gelb
- Note besser als 3,5: grün

```
if (mark[position].equals("5,0" ))
 holder.tv_mark.setTextColor
  (context.getResources().getColor(R.color.magenta));
if (mark[position].equals("4,0" ) |
    mark[position].equals("3,9" ) |
    mark[position].equals("3,8" ) |
    mark[position].equals("3,7" ) |
    mark[position].equals("3,6" ) |
    mark[position].equals("3,5" ) )
         holder.tv_mark.setTextColor
          (context.getResources().getColor(R.color.gelb));
    else
         holder.tv_mark.setTextColor
          (context.getResources().getColor(R.color.gruen));
```

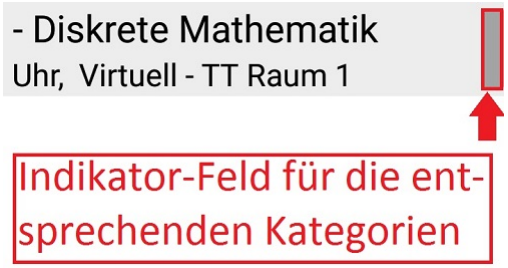
### 6.7.4 CustomAdapterStundenplan

Dieser CustomAdapter befüllt die ListView des StundenPlanfragments.
Hier werden fünf StringArrays(datum,fach,zeit,raum,kategorie) übergeben, die zuvor mittels der Methode *erstelleStundenplan()* des StundenplanFragments aus dem HTML-Code der QiS/HiS-Seite ausgelesen wurden.
Auch hier gibt es ein einige spezifische Formatierungen, abhängig vom übergebenen Inhalt.
Ist der Inhalt des StringArrays datum an einer Stelle "NULL", so wird die Sichtbarkeit des zugehörigen TextViews auf "GONE" gesetzt.
Analog zum *CustomAdapterNoten* werden so die einzelnen Fächer unter dem selben Datum gelistet. Andernfalls würde über jedem Fach ein dunkelgraues TextView stehen.

```
if(datum[position]!=null)
   holder.tv_date.setText(datum[position]);
else
   holder.tv_date.setVisibility(TextView.GONE);
```

Das StringArray kategorie wird dazu verwendet, wichtige Ereignisse farblich kenntlich zu machen. Dabei wird nur der Inhalt an der jeweiligen Position des StringArrays geprüft und eine entsprechende (CI/CD-)Farbe für das Rechteck auf der rechten Seite verwendet:

- Prüfung: magenta
- Praktikum: dunkelblau
- Rest: grau01




```
if (kategorie[position].equals("Pruefung"))
   holder.tv_category.setBackgroundColor
   (context.getRessources().getColor(R.color.magenta));
else if ((kategorie[position].equals("Praktikum"))
   holder.tv_category.setBackgroundColor
   (context.getRessources().getColor(R.color.dunkelblau));
else
    holder.tv_category.setBackgroundColor
   (context.getRessources().getColor(R.color.grau01));
```