Fachbereich Mathematik / Informatik

Konzeption einer plattformunabhängigen Applikation für Tablett-Systeme und deren Umsetzung am Beispiel eines digitalen Klassenbuchs

von: Björn Hinze

Matrikelnummer: 1898193

Gutachter: Prof. Dr. Karl-Heinz Rödiger

Gutachter: Dr. Dieter Müller

Bremen den 30. Juni 2012

1 EINLEITUNG 5

1.1 ZIEL DER ARBEIT 5

1.2 MOTIVATION UND AUSGANGSSITUATION 5

1.3 AUFBAU DER ARBEIT 10

2 KONZEPTION PLATTFORMUNABHÄNGIGER APPLIKATIONEN 12

2.1 PLATTFORMUNABHÄNGIGKEIT 12

2.1.1 Crosscompiler 12

2.1.2 Virtual-Machine 12

2.1.3 Hybrid-Applikationen 13

2.2 AUSWAHLKRITERIEN 13

2.2.1 Titanium 14

2.2.2 PhoneGap 15

2.2.3 ApplicationCraft 16

2.2.4 Zusammenfassung 17

2.3 RICHTLINIEN FÜR BENUTZUNGSOBERFLÄCHEN 17

2.3.1 Vergleich der nativen visuellen Elemente 19

2.3.2 Umsetzung der Elemente mit Hilfe des DojoToolkits 21

2.3.3 Implementierung fehlender Elemente 23

2.3.3.1 Toolbar 24

2.3.3.2 Popover-View 27

2.3.3.3 Alert 30

2.3.3.4 Action Sheet 30

2.3.3.5 Modal View 31

2.3.3.6 Activity Indicator 32

2.3.3.7 Detail Disclosure Button 33

2.3.3.8 Info Button 33

2.3.3.9 Search Bar 33

2.4 ZUSAMMENFASSUNG 35

3 RAHMENBEDINGUNGEN FÜR EIN DIGITALES KLASSENBUCH 36

3.1 DER BILDUNGSPOLITISCHE AUFTRAG 36

3.2 DIE AUSGANGSSITUATION 39

3.2.1 Die schulorganisatorische Rahmenbedingungen 40

3.2.1.1 Das aktuelle Klassenbuch 41

3.2.1.2 Anforderungen an das elektronische Klassenbuch 42

3.2.2 Die unterrichtsorganisatorischen Rahmenbedingungen 44

3.2.3 Die bildungspolitischen Rahmenbedingungen (Inklusive Beschulung) 44
3.2.4 Die rechtlichen Rahmenbedingungen (Datenschutz) 46
3.2.5 Die technischen Rahmenbedingungen 48
3.2.5.1 Netzkonzept 48
3.2.5.2 Schulische Ausstattung 49
3.2.5.3 Sicherheitskonzept 50
3.2.5.4 Datenbankbeschreibung 51
3.3 ZUSAMMENFASSUNG 51

4 UMSETZUNG EINES DIGITALEN KLASSENBUCHES 53
4.1 FUNKTIONSSPEZIFIKATION 53
4.2 DATENVERWALTUNG 54
4.2.1 Konzeption der Datenbank 54
4.2.2 Asynchrone Datenübertragung 62
4.3 BESCHREIBUNG DER BENUTZUNGSOBERFLÄCHE 64
4.3.1.1 Login-Screen 65
4.3.1.2 Stundenplan-Screen 66
4.3.1.3 Stundenverwaltungs-Screen 67
4.3.1.3.1 Fehlzeitenerfassung 67
4.3.1.3.2 Schülerübersicht 68
4.3.1.3.3 Thema der Stunde 69
4.3.1.4 ModalViews 69
4.3.1.5 Hilfe und Optionen 70
4.4 DIE IMPLEMENTIERUNG 71
4.4.1 Die Entwicklungsumgebungen 71
4.4.2 Das DojoToolkit 71
4.4.3 Die Implementierung ausgewählter Funktionen 73
4.4.3.1 Die Synchronisation mit dem Schulserver 74
4.4.3.2 Die Speicherung in der lokalen Datenbank 76
4.4.3.3 Das Einloggen 77
4.4.3.4 Der dynamische Stundenplanaufbau 77
4.4.3.5 Die Fehlzeitenerfassung 78
4.4.3.6 Die Schülerübersicht 79
4.4.3.7 Die Stundenübersicht 79
4.5 ZUSAMMENFASSUNG 80

5 AUSBLICK UND FAZIT 81
5.1 AUSBLICK 81
5.2 FAZIT 83

6 ABBILDUNGSVERZEICHNIS 85

7 LITERATURVERZEICHNIS 87

1 Einleitung

Immer mehr Aufgaben, die vor kurzem noch an einem klassischen, stationären Computer erledigt wurden, werden mittlerweile verstärkt mit Hilfe mobiler Geräte bearbeitet. Dabei sind nach Notebooks, Netbooks und Smartphones neuerdings auch sogenannte Tablett-Systeme verstärkt im Einsatz. So gewinnt die Entwicklung geeigneter Anwendungen für genau diese Systemarchitekturen immer mehr an Relevanz und stellt neben dem wirtschaftlichen Aspekt auch eine informationstechnische Herausforderung dar.

Im Folgenden werden nun die Ziele und Aufgaben dieser Arbeit näher beschrieben. Der Lesbarkeit halber wird in der gesamten Arbeit die maskuline Form gewählt, es sind jedoch immer beide Geschlechter gemeint.

1.1 Ziel der Arbeit

Um der beschriebenen Entwicklung Rechnung zu tragen, soll in dieser Arbeit ein Konzept erstellt werden, mit dem plattformunabhängige Applikationen für Tablett-Systeme entwickelt werden können. Als Ausgangspunkt steht dabei folgende Frage: Inwiefern ist es möglich native Applikationen mit einem plattformunabhängigen Ansatz zu emulieren? Die These, dass ein solches Vorhaben generell möglich ist, soll anhand einer konkreten Umsetzung überprüft werden. Die Erstellung eines digitalen Klassenbuchs dient hier als Fallbeispiel.

1.2 Motivation und Ausgangssituation

Im klassischen PC-Bereich haben sich drei wichtige Plattformen Windows, Linux und Mac OSX für die kommerzielle Anwendungsentwicklung herauskristallisiert. Bei mobilen Geräten konkurrieren derzeit (2012) deutlich mehr Betriebssysteme. So finden sich auf Smartphones neben iOS und Android fünf weitere Plattformen (WindowsPhone, Blackberry, WebOS, Symbian, Bada), die prinzipiell auch auf Tablett-Systemen zum Einsatz kommen könnten.

Aufgrund der derzeitigen Marktsituation (Ende 2011) stellt sich im Bereich der Tablett-Systeme die Situation deutlich weniger vielfältig dar.

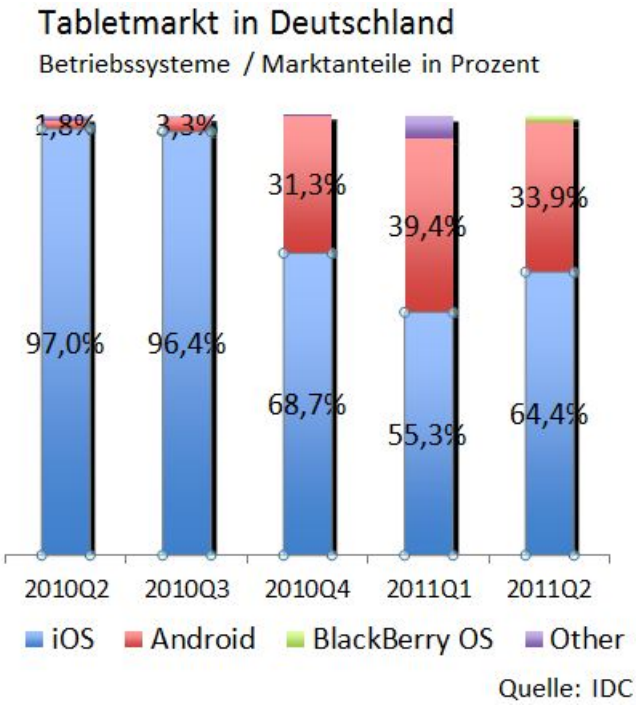

Abbildung 1: Tablett-Systeme Marktanteile Deutschland (FAZ, 2011)

Wie in Abbildung 1: Tablett-Systeme Marktanteile Deutschland ersichtlich, wird der Markt durch die beiden Betriebssysteme iOS und Android dominiert. Es entfallen im zweiten Quartal 2011 64,4% der Marktanteile auf iPad/iPad2 und 33,9% auf Tablett-Systeme mit Android. Die übrigen 1,7% verteilen sich auf alle anderen Betriebssysteme wie zum Beispiel Windows mobil, WebOS oder Blackberry. Zwar wird der Markt derzeit durch iOS und Android beherrscht, doch auch RIM (Blackberry) und Microsoft versuchen Marktanteile zu gewinnen beziehungsweise zu halten. Durch die Vielfalt mobiler Plattformen kommt dem Konzept einer plattformübergreifenden Anwendungsentwicklung eine besondere Bedeutung zu.

Bereits seit 2008 begann die Entwicklung derartiger Frameworks, von denen heute mehr als dreißig mit unterschiedlichen Leistungsmerkmalen verfügbar sind (Falk, 2012).

Für eine Nutzung der gerätespezifischen, speziell auf das Betriebssystem ausgerichteten Anwendungen, werden sogenannte native Applikationen eingesetzt. Dabei stehen für die jeweilige Plattform ein entsprechendes *Software Development*

Kit (SDK) und eine Entwicklungsumgebung zur Verfügung. Programmiert wird in der von der Plattform vorgegebenen Sprache. Wie die folgende Tabelle zeigt, wird zum Beispiel für Android in Java und für iOS in Objective-C gearbeitet.

Plattform	**Android**	**iOS**	**Windows Phone 7**	**Symbian**	**Blackberry OS**
Sprache	Java	Objective-C	C#	C++	Java

Abbildung 2: Smartphone Plattformen mit dazugehörigen Programmiersprachen

Nahezu jede dieser Plattformen erfordert eine besondere Programmiersprache. Dies führt im Ergebnis dazu, dass je nach Hersteller und sogar Betriebssystemversion, spezielle und in der Regel zueinander inkompatible Anwendungen erstellt werden müssen.

Hierdurch wird der hohe Aufwand der Softwareentwicklung verdeutlicht, welcher betrieben werden muss, wenn für verschiedene mobile Betriebssysteme ein und dieselbe native Anwendung erstellt werden soll. Eine der Grundvoraussetzungen stellt das fundierte Wissen in den verschiedenen Programmiersprachen. Die proprietären Betriebssysteme der mobilen Endgeräte erfordern für jedes Gerät bei gleicher Aufgabenstellung (zum Beispiel E-Mails abrufen) eine spezielle, native Implementierung der Applikation, um den vollen Funktionsumfang des Betriebssystems in Verbindung mit den Hardwarekomponenten zu ermöglichen. Der betriebene Aufwand zur Erstellung und Pflege des Programmcodes wird durch die Anzahl der zu unterstützenden Betriebssysteme erheblich erhöht. Der Vorteil von nativen Anwendungen besteht allerdings in der Performance bei der Nutzung der gerätespezifischen Hardware durch die optimale Anpassung an das Betriebssystem. Im Falle des iOS ist dies in der Abbildung 3: Programmfluss der nativen iOS Benutzungsoberfläche schematisch dargestellt.

„The Cocoa Touch frameworks that drive iOS apps share many proven patterns found on the Mac, but were built with a special focus on touch-based interfaces and optimization” (Apple Inc).

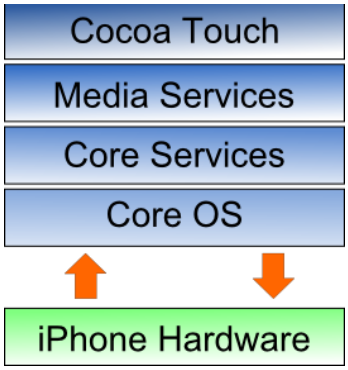

Abbildung 3: Programmfluss der nativen iOS Benutzungsoberfläche (Apple Inc)

Der entsprechende Aufbau des Android-Systems ist in Abbildung 4: Aufbau des Android Betriebssystems dargestellt.

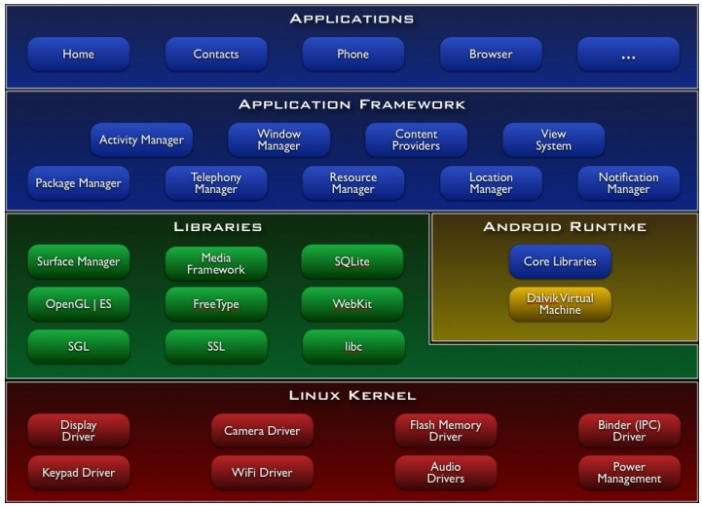

Abbildung 4: Aufbau des Android Betriebssystems (Google Inc., 2012)

"Android is a layered environment built upon a foundation of the Linux kernel, and it includes rich functions. The UI subsystem includes:

- Windows
- Views
- Widgets

for displaying common elements such as edit boxes, lists, and drop-down lists" (Google Inc., 2012).

Dieser modulare Aufbau der Betriebssysteme hat Konsequenzen für die Softwareentwicklung.

"There are two types of software that you can develop for iOS-based devices:

- iOS apps
- Web content
- An **iOS app** is an application you develop using the iOS SDK to run natively on iOS-based devices. iOS apps resemble the built-in applications on iOS-based devices in that they reside on the device itself and take advantage of features of the iOS environment. People install iOS apps on their devices and use them just as they use built-in applications, such as Photos, Calendar, and Mail.
- **Web content** is hosted by a website that people visit through their iOS-based devices.

  [...]
- Webpages […] operate as designed (with the exception of any elements that rely on unsupported technologies, such as plug-ins, Flash, and Java)" (Apple Inc., 2011, S. 17).

Neben diesen Beschränkungen im Falle des iOS ist der direkte Zugriff auf hardwarespezifische Ressourcen der mobilen Geräte wie zum Beispiel Kamera, lokaler Speicher und GPS auch für moderne Webbrowser aus diesen Web-Applikationen nicht direkt möglich.

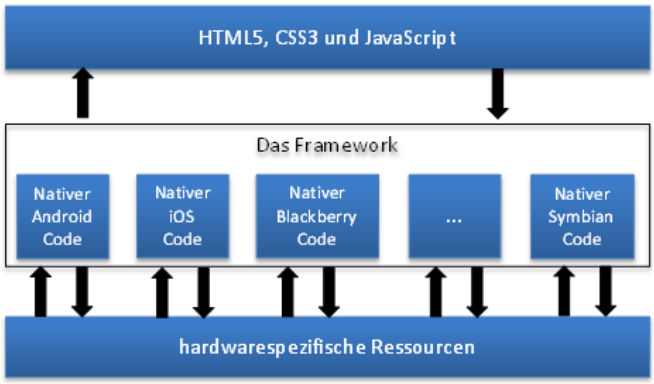

Abbildung 5: Konzept für ein Plattformunabhängiges Framework

Allerdings kann der Zugriff auf die Gerätehardware prinzipiell mit Hilfe geeigneter Frameworks realisiert werden, die einerseits, eine gemeinsame, für alle Plattformen gleiche Sprache zur Verfügung stellt, darüber hinaus aber auch über *Application Programming Interfaces* (APIs) des jeweiligen Betriebssystem für den Zugriff auf die Hardware verfügen. Die auf mobilen Geräten vorhandenen modernen Webbrowser bieten mit *Hypertext Markup Language* (HTML5), JavaScript und *Cascading Style Sheets* (CSS3) zahlreiche neue Möglichkeiten für Entwickler mit Web-App‘s standard-konforme Anwendungen zu entwickeln und damit den Großteil der mobilen Plattformen zu erreichen. Die Webtechnologie bietet damit eine Möglichkeit für alle Betriebssysteme in gleicher Sprache zu entwickeln. Applikationen die einen solchen Ansatz verfolgen werden als Hybrid-Web-Apps bezeichnet. Die Herausforderung dieser Arbeit ist es, mit dem digitalen Klassenbuch eine Anwendung zu erstellen, die diesem Ansatz folgt und qualitativ Gleichwertig gegenüber einer nativen Applikationen ist.

1.3 Aufbau der Arbeit

Zu Beginn gilt es anhand von geeigneten Kriterien (s. Kapitel 2.2) ein Framework auszuwählen, das eine geräteunabhängige Schnittstelle für den Zugriff auf hardwarespezifische Ressourcen ermöglicht. Die mit Hilfe dieses Frameworks aus einer gemeinsamen Codebasis generierte Anwendung für ein iPad und ein Android-Tablett sollen sich von nativen Applikationen für die jeweilige Plattform nicht wesentlich unterscheiden. Dieses gilt insbesondere für die Handhabung und optische

Darstellung. Ob diese Vorgehensweise jedoch eine echte Alternative zur nativen Anwendungsentwicklung darstellt, wird im Rahmen dieser Arbeit abschließend untersucht und bewertet.

Weiterhin erscheint es zweckmäßig für die Entwicklung der Benutzungsoberfläche eine Softwarebibliothek zu benutzen, die durch geeignete Schnittstellen die Darstellung der typischen Oberflächen ermöglicht. Durch eine Gegenüberstellung der beiden Style-Guides soll herausgearbeitet werden, ob signifikante Unterschiede eine plattformunabhängige Programmentwicklung verhindern oder ob bei konsequenter Berücksichtigung diese Art der Anwendungsentwicklung eine geeignete Alternative zur nativen Programmierung ist (vgl. (Apple Inc., 2011); (Google Inc., 2011)). Am Beispiel der konkreten Applikation für den Schulbereich sollen diese Thesen auf ihre Praxistauglichkeit überprüft werden.

2 Konzeption plattformunabhängiger Applikationen

Um aus der Vielzahl der am Markt angebotenen Frameworks eine Auswahl zu treffen, werden hier Merkmale gegenübergestellt und diskutiert. Ergänzend zu dieser Betrachtung werden die Style-Guidelines von Apple mit denen von Google verglichen und hinsichtlich ihrer Realisierung im DojoToolkit überprüft. Die bei der Analyse festgestellten Lücken werden beschrieben und durch eigene Entwicklungen ergänzt.

2.1 Plattformunabhängigkeit

Es existieren unterschiedliche Ansätze plattformunabhängige Applikationen für mobile Endgeräte zu entwickeln, welche im Einzelnen dargestellt und diskutiert werden. Den Abschluss dieses Kapitels bildet die Auswahl des geeigneten Systems, das die angelegten Kriterien erfüllt.

2.1.1 Crosscompiler

Dieser Lösungsansatz abstrahiert für die Programmentwicklung den Programmcode vollständig von den Besonderheiten der Zielsysteme. Hierfür stellt die Entwicklungsumgebung eine plattformunabhängige Schnittstelle in einer der für mobile Systeme gängigen Programmiersprachen (Java, JavaScript oder Ruby) zur Verfügung. Es werden die Elemente der Benutzungsoberfläche, die Programmlogik und die Datenstrukturen definiert und über einen Cross-Plattform-Compiler in eine plattformspezifische native App übersetzt. Diese kann dann auf die Zielsysteme verteilt und dort ausgeführt werden.

Der Vorteil dieser Lösung besteht in der Konsistenz der Benutzungsoberfläche der Zielsysteme und der Performance, während der Nachteil in der Komplexität der Entwicklung der Compiler liegt. Daher handelt es sich in der Regel um kommerzielle Plattformen, die diesen Ansatz verfolgen.

2.1.2 Virtual-Machine

Eine dem Crosscompiler ähnliche Lösung wird durch die Abstraktion der Anwendungsebene von den Besonderheiten des Zielsystems durch die Nutzung einer virtuellen Maschine realisiert. Hierfür stellt die Programmierumgebung die jeweilige virtuelle Maschine zur Verfügung. Dieser Lösungsansatz wird unter anderem von der Plattform Rhodes gewählt (vgl. Rhomobile, 2011).

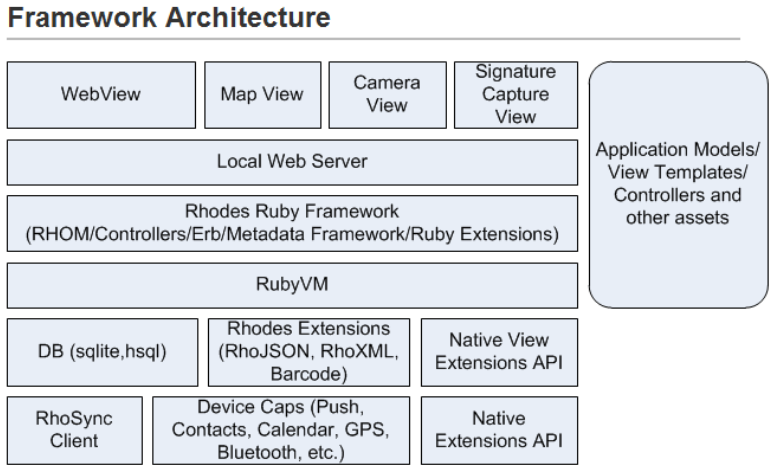

Abbildung 6: Aufbau des Rhodes-Framework (Rhomobile, 2011)

2.1.3 Hybrid-Applikationen

Dieser Ansatz zeichnet sich dadurch aus, dass zur Entwicklung der Benutzungsoberfläche und der Anwendungslogik Standard-Web-Technologien wie HTML5, CSS3 und JavaScript genutzt werden. Durch entsprechende Bibliotheken kann unter anderem auf *embedded-SQL-Datenbanken* (SQLite), lokalen Speicherplatz oder das WLAN zugegriffen werden. Hierzu zählen Frameworks wie PhoneGap, Applicationcraft und WebApp-Net, die plattformspezifische Softwaremodule bereitstellen (vgl. u.a. (Nitobi); (Application Craft, 2012); (Apers, 2010)).

2.2 Auswahlkriterien

Als Kriterien für eine Auswahl des im Rahmen dieser Arbeit zu nutzenden Werkzeuges werden angelegt:

- Unterstützung von iOS, Android und Windows als Betriebssystem für Tablett-Systeme
- Lizenzfreie Nutzung des Frameworks zur Entwicklung und Nutzung.
- Unterstützung des Zugriffes auf möglichst viele Hardwareressourcen (mindestens WLAN, Kamera und Speicherzugriff)

Als Vorauswahl wurde die Zusammenstellung von Markus Falk mit Stand vom März 2012 herangezogen (vgl. Falk, 2012). Unter Anlegung der genannten Kriterien konnten aus der Liste der 31 Frameworks lediglich zwei bestehen. Hier handelt es sich um PhoneGap und ApplicationCraft (vgl. (Application Craft, 2012); (Nitobi)). Im Zuge weiterer eigener Recherchen, wurde Titanium gefunden und in den Vergleich auf Grund der Informationen auf den Home-Pages mit einbezogen (vgl. Dalu, 2010).

2.2.1 Titanium

Titanium von Appcelarator Inc. wurde Ende 2008 erstmals freigegeben. Es enthält nicht nur die für eine plattformunabhängige Entwicklung von Tablett-Systemen notwendigen Werkzeuge, sondern adressiert auch Smartphones und Desktop-Systeme. Laut Dokumentation unterstützt das System das komplette *Life-Cycle-Management* (Appcelerator Inc., 2010).

Die Entwicklungsumgebung basiert auf Eclipse und unterstützt, neben der Entwicklung und Evaluierung in den verschiedenen Simulatoren, auch das Deploy der Anwendungen auf die Zielsysteme. Erreicht wird dies durch eine plattformunabhängige JavaScript API, die je nach Bedarf für die verschiedenen Zielsysteme kompiliert werden kann. Während des Kompilierens werden die Besonderheiten des Zielsystems hinsichtlich der benutzten API aufgelöst und die jeweiligen Module eingebunden. Dazu gehören die Benutzungsoberfläche und der Zugriff auf gerätespezifische Treiber. Im letzten Schritt wird vom Compiler eine ausführbare Anwendung generiert.

Obwohl die Entwicklungssprache JavaScript ist, besteht der Unterschied zu den im Folgenden betrachteten Werkzeugen darin, dass für die Anwendung keine Browsertechnologie für das generieren der Benutzungsoberfläche benutzt wird. Stattdessen werden die in der Entwicklungsumgebung abstrahierten Elementtemplates durch native Elemente des Betriebssystems ersetzt. Sollten für den Anwendungsfall die lizenzfreien Core-Module von Titanium ausreichend sein, fallen keine Kosten an. Damit ist Titanium ein vielversprechender Kandidat für eine plattformunabhängige Anwendungsentwicklung, wenn die derzeit fehlende Unterstützung der Windows-Plattform für mobile Geräte ergänzt wird. Die Nutzung

der weiteren Module (Starter- und Enterprisemodule) ist jedoch kostenpflichtig (vgl. Appcelerator Inc., 2010).

2.2.2 PhoneGap

Anfang 2008 erschien die erste Version von PhoneGap. Zur Entwicklung werden ausschließlich HTML5, JavaScript und CSS3 eingesetzt. Über eine abstrakte Zwischenschicht wird eine JavaScript-API zur Verfügung gestellt mit der unter anderem Speicher, Datenbank, Kamera, sowie weitere Funktionen des Betriebssystems angesprochen werden können. Zur Gestaltung der Benutzungsoberfläche ist es allerdings sinnvoll zusätzliche JavaScript-Bibliotheken wie Sencha Touch, JQuery oder wie auch im Rahmen dieser Arbeit das DojoToolkit einzubinden.

Abbildung 7: Hardwareunterstützung von mobilen Geräten (Nitobi)

2.2.3 ApplicationCraft

ApplicationCraft bietet eine hundertprozentige cloudbasierte Entwicklungsumgebung zur Erstellung von Anwendungen für Smartphones, Tabletts und Desktop-Systemen. Hierfür wird ein WYSIWYG-Editor genutzt, in dem per Drag & Drop eine Benutzungsoberfläche erstellt werden kann. Um mit diesem Werkzeug style-guide konforme Anwendungen erstellen zu können, reichen die mit Hilfe des WYSIWYG-Editors bereitgestellten Widgets nicht aus. Darüber hinaus stellt der in der kostenfreien Version von ApplicationCraft verfügbare Funktionsumfang keine Alternative für eine professionelle Anwendungsentwicklung dar. Es eignet sich allenfalls für die unkomplizierte Erstellung eines Prototyps als Grundlage für eine anschließende Anwendungsentwicklung.

Inwiefern die kostenpflichtige Variante die geforderten Kriterien erfüllen würde, kann auf Grundlage der auf der Home-Page gefundenen Informationen nicht abschließend beantwortet werden (vgl. Application Craft, 2012).

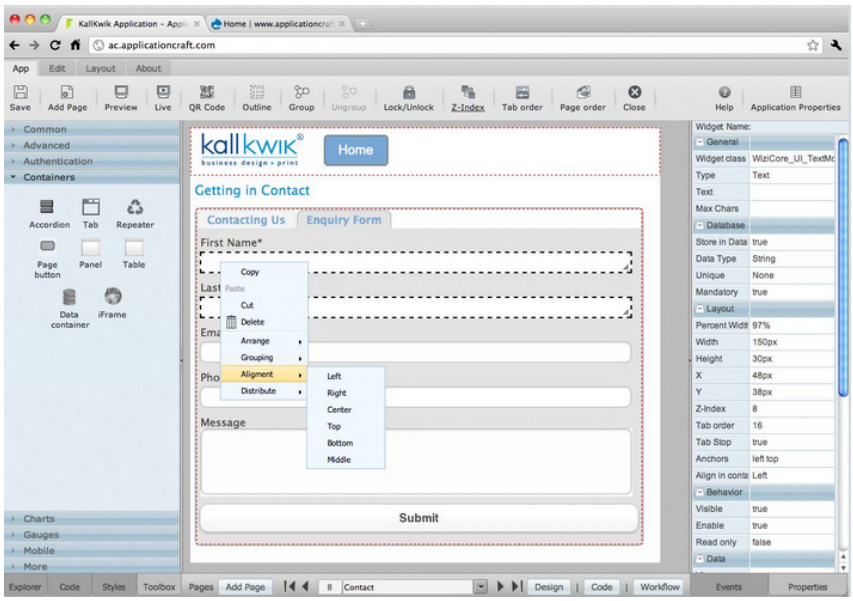

Abbildung 8: Entwicklungswerkzeug von Application Craft (Application Craft, 2012)

2.2.4 Zusammenfassung

Eine abschließende Bewertung der betrachteten Entwicklungswerkzeuge wird durch die hohe Dynamik der Weiterentwicklung erschwert. Allein während des Zeitraums der Erstellung dieser Arbeit wurde PhoneGap von der Version 0.9.3 in mehreren Schritten auf 1.8.0 aktualisiert. PhoneGap ist derzeit das einzige Entwicklungswerkzeug, welches die in Abschnitt 2.2 genannten Kriterien weitgehend erfüllt. Ausschlaggebend ist dabei die Möglichkeit, die Benutzungsoberfläche nach Belieben zu gestalten und damit die Entwicklung einer Style-Guide konformen Benutzungsoberfläche zu gewährleisten.

2.3 Richtlinien für Benutzungsoberflächen

Für die Akzeptanz einer mobilen Applikation ist die Gestaltung der Benutzungsoberfläche von besonderer Bedeutung. Dabei spielen die Aspekte der Benutzungsfreundlichkeit wie Einfachheit, Effektivität, Intuitivität (Erwartungskonformität), Fehlertoleranz und einheitliche Gestaltung eine entscheidende Rolle.

„A great user interface follows human interface design principles that are based on the way people—users—think and work, not on the capabilities of the device. A user interface that is unattractive, convoluted, or illogical can make even a great application seem like a chore to use. But a beautiful, intuitive, compelling user interface enhances an application’s functionality and inspires a positive emotional attachment in users” (Apple Inc., 2011, S. 21).

Diese Philosophie spiegelt sich ebenfalls im Android-Design wieder.

“We focused the design work with three overarching goals for our core apps and the system at large. As you design apps to work with Android, consider these goals: Enchant me, Simplify my life, Make me amazing” (Google Inc., 2011).

Hinweise für die Entwicklung von Anwendungen für das Android-Betriebssystem sind Anfang Januar im Zusammenhang mit der Freigabe der Version 4 konkretisiert worden.

Für eine plattformunabhängige Anwendungsentwicklung ist es wichtig, die Handhabung und optische Darstellung der Zielsysteme zu beachten. Die Unterschiede können aus den jeweiligen Style-Guides abgeleitet werden. Im iOS Style-Guide ist das komplette Design genau einer Benutzungsoberfläche beschrieben, während für Android drei Design-Vorlagen *Holo Light*, *Holo Dark* und *Holo Light with dark action bars* aufgeführt werden (Google Inc., 2011).

Finden sich im Android Design einige generelle Festlegungen für das Touch-Feedback, so ist hierfür im iOS kein Konzept explizit beschrieben. Ebenso werden die System Fonts im iOS (Helvetica New) in den Style-Guides von Apple nicht erwähnt, während im Android Design besonders auf die Nutzung der Schriften *Roboto Regular* und *Roboto Bold* verwiesen wird. Für die Gestaltung von Elementen, die nicht im Umfang des jeweiligen Betriebssystems definiert sind, gibt Google im Gegensatz zu Apple genaue Farbvorgaben.

„Blue is the standard accent color in Android's color palette“ (Google Inc., 2011).

Einer der zentralen Punkte stellt die Nutzung der grafischen Elemente durch Gesten oder Tippen mit den Fingern dar. Damit kommt der Festlegung der Größe von grafischen Elementen eine besondere Bedeutung zu. Extrem kleine Interaktionsflächen, welche bei einer einfachen Portierung von Webseiten aus den üblichen HTML-Elementen wie zum Beispiel einer Checkbox leicht entstehen können verhindern eine ergonomische Bedienung.

“The comfortable minimum size of tappable UI elements is 44 x 44 points” (Apple Inc., 2011, S. 13).

Daher wird auch für Android die Mindestgröße von grafischen Elementen vorgegeben.

„On average, 48dp translate to a physical size of about 9mm (with some variability). This is comfortably in the range of recommended target sizes (7-10 mm) for touchscreen objects and users will be able to reliably and accurately target them with their fingers” (Google Inc., 2011).

Durch die grundsätzlich verschiedene Handhabung gegenüber der Nutzung von Programmen mit Maus und Tastatur, kommt der Style-Guide konformen Entwicklung

grafischer Elemente eine besondere Bedeutung zu. Als Referenz für eine Benutzungsoberfläche werden im Rahmen dieser Arbeit die Richtlinien für die vorbildlich ergonomischen visuellen Elemente von Apple angelegt. Diese werden in Abschnitt 2.3.1 den Elementen des Android-Betriebssystems gegenübergestellt. Dabei gelten die iOS–Guidelines vom 23.3.2011 von Apple und die Style-Guides für Android vom Januar 2012.

2.3.1 Vergleich der nativen visuellen Elemente

In der folgenden Tabelle werden die jeweiligen Objekte gegenübergestellt und entsprechend den Hinweisen in der Spalte *Anmerkung* analysiert und kommentiert

iOS Bezeichnung	iOS Style-Guide Seitenzahl	Android Bezeichnung	Unterrubrik im Android Design	Anmerkung
Navigation Bar	98	View Control	UI Overview	Android kann Navigationsliste und Tab Bar enthalten. Dafür hat der Back-Btn kein Label. Im iOS ist dies nicht gestattet.
Toolbar	100	Split Action Bar	UI Overview	gleich
Tab Bar	101	Tabs, Top bar, Fixed Tabs	App Structure, Action Bar Tabs	gleich aber: Android oben iOS unten
Popover View	103	Popups	Dialogs	gleich
Split View	105	single compound view	Multi-pane Layouts	gleich
Table View	107	master grid oder list view, detail View, Lists, Grid List	Multi-pane Layouts, Lists, Grid Lists	gleich
Text View	114	Content Area	Common App UI	gleich
Web View	115	Content Area	Common App UI	gleich
Alert	116	Alert	Dialods	gleich
Action Sheet	119	share action provider,	App structure, Action Bar,	gleich

		Spinners	Spinners	
Modal View	121	Dialogs	Dialogs	gleich
Activity Indicator	124	Activity circle	Feedback	gleich
Date and Time Picker	125	Picker	Picker	gleich
Detail Disclosure Button	126			In Android nicht spezifiziert, aber leicht umsetzbar
Info Button	126			In Android nicht spezifiziert, aber leicht umsetzbar
Label	127			In Android nicht spezifiziert, aber leicht umsetzbar
Page Indicator	128			In Android nicht besonders spezifiziert; wird üblicherweise textuell durch z.B.: *5 von 12* umgesetzt
Picker	129	Pickers	Pickers	gleich
Progress View	129	Progress, Activity	Feedback	gleich
Rounded Rectangle Button	130	Buttons	Buttons	gleich
Scope Bar	131			In Android nicht spezifiziert, aber leicht umsetzbar
Search Bar	131			In Android nicht spezifiziert, aber leicht umsetzbar
Segmented Control	132	drop-downs or split list items	Action Bar	gleich
Slider	133	Seek Bars and Sliders	Seek Bars and Sliders	gleich
Switch	134	Switches	Switches	gleich
Text Field	134	Text fields	Text fields	gleich
System-Provided Buttons and	135	Action buttons	Action Bar, Pure Android	gleich

Icons				
Views & Swipegeste	115, 114, 105, 107 und 15	Swipe View	App structure	gleich
		Back Button	Up vs. Back	nicht umgesetzt
		Scrollable Tabs, Stacked Tabs	Tabs	Im iOS nicht erlaubt.

Abbildung 9: Vergleich der nativen Elemente des iOS mit Android

Zusammenfassend lässt sich feststellen, dass mit Ausnahme der drei Android-Elemente *Hardware-Back-Button* und *Scrollable*- beziehungsweise *Stackabletabs*, eine plattformunabhängige Implementierung möglich erscheint. Wenn auf diese Elemente verzichtet werden kann und die Positionierung der Tab Bar berücksichtigt wird, die im iOS den unteren Abschluss des Bildschirmes bildet, während im Android-OS diese den obersten Teil des Bildschirms darstellt, so ist festzuhalten, dass zwar keine Gleichheit aber eine weitgehende Entsprechung der Funktionalität der nativen visuellen Elemente besteht. Daher wird im Folgenden die Implementierung der nativen Elemente mit Hilfe geeigneter plattformunabhängiger Software-Bibliotheken untersucht.

2.3.2 Umsetzung der Elemente mit Hilfe des DojoToolkits

PhoneGap kann durch HTML5, CSS3 und JavaScript-Bibliotheken hinsichtlich des Funktionsumfanges und der Darstellung der Benutzungsoberfläche erweitert werden. Keine der alternativ betrachteten Bibliotheken, wie SenchaTouch oder JQueryMobile, bietet zum jetzigen Zeitpunkt diesen Umfang und die optische Qualität des DojoToolkits. Konzeptionell wird durch Austausch der CSS3-Datei für das jeweilige grafische Element die Darstellung konform zum Zielsystem realisiert. So finden sich CSS-Dateien für iPhone, iPad, Android und Blackberry im Toolkit. Durch Abfrage des Zielsystems zur Laufzeit können die entsprechenden CSS-Dateien eingebunden werden (vgl. The Dojo Foundation, 2012).

Da aus Abbildung 9: Vergleich der nativen Elemente des iOS mit Android hervorgeht, dass die Elemente der Benutzungsoberfläche von Apple bis auf die drei

kommentierten Ausnahmen eine Entsprechung im Android-System finden, werden im Folgenden lediglich die iOS-Elemente bezüglich ihrer entsprechende Realisierungen mit Hilfe des DojoToolkits betrachtet und kommentiert.

Element-Nr	iOS Guidelines Seitenzahl	Apple Bezeichnung	DojoToolkit Bezeichnung	Richtlinie erfüllt
I	98	Navigation Bar	dojox.mobile.Heading dojox.mobile.ToolBarButton	Ja
II	100	Toolbar		Nein
III	101	Tab Bar	dojox.mobile.TabBar dojox.mobile.TabBarButton	Ja
IV	103	Popover View	Dojox.mobile.Opener	Ja, aber z.Zt. fehlerhaft
V	105	Split View	dojox.mobile.fixedSplitter dojox.mobile.fixedSplitterPane	Ja
VI	107	Table View	dojox.mobile.EdgeToEdgeList dojox.mobile.EdgeToEdgeCategory dojox.mobile.ListItem dojox.mobile.RoundRect dojox.mobile.ScrollableView	Ja, aber Symbole müssen angepasst werden
VII	114	Text View	dojox.mobile.TextArea	Ja
VIII	115	Web View	dojox.mobile.View	Ja
IX	116	Alert		Nein
X	119	Action Sheet		Nein
XI	121	Modal View		Nein
XII	124	Activity Indicator		Nein
XIII	125	Date and Time Picker	dojox.mobile.SpinWheelDatePicker	Ja
XIV	126	Detail Disclosure Button		Nein
XV	126	Info Button		Nein

XVI	127	Label	Default Text in einer View	Ja
XVII	128	Page Indicator	dojox.mobile.Carousel	Ja
XVIII	129	Picker	dojox.mobile.SpinWheel	Ja
XIX	129	Progress View	dijit.ProgressBar	Ja
XX	130	Rounded Rectangle Button	dojox.mobile.Button	Ja
XXI	131	Scope Bar	dojox.mobile.Heading dojox.mobile.TabBar dojox.mobile.TabBarButton	Ja
XXII	131	Search Bar		Nein
XXIII	132	Segmented Control	dojox.mobile.TabBar dojox.mobile.TabBarButton	Ja
XXIV	133	Slider	dojox.mobile.Slider	Ja
XXV	134	Switch	dojox.mobile.Switch	Ja
XXVI	134	Text Field	dojox.mobile.TextBox	Ja
XXVII	135	System-Provided Buttons and Icons	dojox.mobile.Button dojox.mobile.TabBarButton dojox.mobile.ToolBarButton	Nein

Abbildung 10: Realisierbarkeit der nativen iOS Elemente mit dem DojoToolkit

Das Fehlen einer Toolbar oder eines Popover Views stellt in der aktuellen Version von DojoToolkit eine Einschränkungen dar, die aber durch den konzeptionellen Aufbau des DojoToolkits begünstigt, durch eine eigene Implementierung ausgeglichen werden kann.

2.3.3 Implementierung fehlender Elemente

Da am 8. April 2012 die UX-Richtlinien für Windows-Apps im Metro-Stil noch immer vorläufig waren, wurde auf eine explizite Adaption verzichtet, zumal Dojo zum jetzigen Zeitpunkt Windows nicht unterstützt (vgl. Microsoft, 2012).

„Dojo is also tested with popular mobile browsers including iOS 4.x and 5.x, Android 2.x, 3.x, and 4.x, and Blackberry 6, and passes for all supported features in Dojo Mobile, and most features throughout Dojo. Work is planned for supporting Blackberry 7 and QNX, and Windows Phone 7.5“ (The Dojo Foundation, 2011).

Um sicher zu gehen, dass eine plattformunabhängige Implementierung einer Benutzungsoberfläche für mindestens die Zielsysteme iOS und Android konzeptionell möglich ist, wurden die neun fehlenden Elemente *Tool Bar*, *Popover View*, *Alert*, *Action Sheet*, *Modal View*, *Activity Indicator*, *Detail Disclosure Button*, *Info Button* und *Search Bar* in einer eigenen Implementierung realisiert. Dabei wurde besonderer Wert auf eine allgemeingültige Implementierung gelegt.

Um auf diese Elemente später zugreifen zu können, müssen die Dateien

- additionalUIElements.js
- css/themes/iphone/additionalUIElements.css
  oder
- css/themes/android/additionalUIElements.css

in die Applikation eingebunden werden.

Im Folgenden werden die Funktionalität und die Umsetzung der einzelnen fehlenden Elemente dargestellt.

2.3.3.1 Toolbar

„On iPhone, a toolbar always appears at the bottom edge of a screen or view, but on iPad it can instead appear at the top edge.

Toolbar items are displayed equally spaced across the width of the toolbar. The precise set of toolbar items can change from view to view, because the items are always specific to the context of the current view.

On iPhone, changing the device orientation from portrait to landscape can change the height of the toolbar automatically. On iPad, the height and translucency of a toolbar does not change with rotation“ (Apple Inc., 2011, S. 100).

Um diese Anforderungen zu erfüllen, wurden für die Entwicklung die Standarticons im Portable Network Graphics (Png) Format gespeichert.

Zur Erstellung einer Tool Bar sollte ein *dojox.mobile.Header* als Elternelement genutzt werden. Eine Möglichkeit, ein solches Objekt zu erstellen, wurde in der JavaScript-Funktion *createToolBar* umgesetzt, die in der Datei *additonalUIElements.js* enthalten ist. Hierbei wird ein assoziatives Array als

Übergabeparameter verwendet. Bei erfolgreicher Erzeugung der Tool Bar wird diese als HTML-Element zurückgegeben. Zulässige Parameter des assoziativen Arrays sind:

Bezeichnung	Value	Beschreibung
parentId	HTML id	Zu Übergeben ist die ID des Eltern-Elements in welches die Tool Bar eingefügt werden soll
tools	String Array	Für jeden Button, der in der Tool Bar angezeigt werden soll, muss das Element im Array über sein Schlüsselwort definiert werden
Schlüsselwörter von tools	customcase action camera compose bookmarks search add trash organize reply stop refresh play fastFroward pause rewind	Dieses Schlüsselwort muss für jedes selbstdefinierte Icon benutzt werden. Gleichzeitig wird hierdurch die Position in der Toolbar festgelegt. Für jedes dieser Icons muss in dem Parameter customIcons die Pfadangabe übergeben werden. Die übrigen Schlüsselworte sind selbsterklärend und werden in Tabelle 7.2 auf Seite 136 (Apple Inc., 2011) mit den jeweiligen System-Icons (system-provided buttons) dargestellt.
customIcons	String Array	Ein Array mit Pfadangaben, wo sich die selbstdefinierten Icons befinden. Dabei steht der i-te Eintrag für das i-te selbstdefinierte Element egal ob zwischen dem i-ten und i+1 noch vordefinierte System-Elemente befinden.

behavior	function Array	Ein Array mit JavaScript Funktionen. Die i-te Funktion wird dem Click-Event des i-ten Tool Bar Element zugeordnet. Elemente ohne Funktion müssen eine leere Funktion oder NULL übergeben bekommen.

Abbildung 11: Parameter der Tool Bar

Abbildung 12: Die iOS Tool Bar (Apple Inc., 2011, S. 100)

Abbildung 13: Die neue Tool Bar

JavaScript-Code der Funktion createToolBar:

```
function createToolBar(probs){
   var el = new dojox.mobile.TabBar();
   el.domNode.style.backgroundColor='transparent';
   el.domNode.style.borderColor='transparent';
   el.domNode.style.backgroundImage='none';
   var j =0;
   var icon1;
   var icon2;
   var slot;
   for(var i=0;i<probs.tools.length;i++){
      switch(probs.tools[i]){
         case 'custom':
            icon1 = probs.customIcons[j].icon1;
            icon2 = probs.customIcons[j].icon2;
            j++;
            break;
         case 'action':
            icon1 = "css/ToolBarAction.png";
            icon2 = "css/ToolBarAction.png";
            break;
               ...
      }
      slot = new dojox.mobile.TabBarButton({'icon1':icon1,'icon2':icon2});
      el.addChild(slot);
      slot.domNode.ontouchend=function(evnt){
         evnt.currentTarget.style.backgroundImage="none";
         evnt.currentTarget.style.backgroundColor="transparent";
      }
      try{
         slot.domNode.ontouchend = probs.behavior[i];
      }
      catch(ex){
         console.error("error in probs.behavior");
      }
   }
   dojo.byId(probs.parentId).appendChild(el.domNode);
```

```
    el.startup();
    return el.domNode;

}
```

2.3.3.2 Popover-View

„A popover is a self-contained view that hovers above the contents of a screen. It always displays an arrow that indicates the point from which it emerged. A popover can contain a wide variety of objects and views [...]“ (Apple Inc., 2011, S. 103).

Dieses Element ist zwar ab der Version 1.7 des Dojox-Modules als *Opener* neu aufgenommen worden und sollte damit auch zur Verfügung stehen. Jedoch stellt sich bei der Nutzung der Funktion heraus, dass diese nicht ausgereift ist. Für Android-Systeme sind diese Ablaufbesonderheiten sogar dokumentiert. In der zukünftigen Version 1.8 sollen die Fehler behoben werden.

„Milestone changed from 1.7.1 to 1.8.

Fix needs some test time before being backported to 1.7“ (doughays, 2011).

Da diese Funktion zurzeit nicht zur Verfügung steht, jedoch im Funktionsumfang unverzichtbar ist, muss eine eigene Implementierung entwickelt werden.

Hierzu wurde für das Objekt ein Basislayout in der CSS-Datei *additionalUIElements.css* erstellt. Für die Erzeugung einer konkreten View steht die JavaScript Funktion *createPopover* aus der JavaScript Datei *additionalUI.js* zur Verfügung. Als Übergabeparameter dient ein assoziatives Array. Die Funktion bietet zwei Alternativen:

Für die erste Variante muss ein HTML-div-Tag mit dem späteren Inhalt definiert werden. Dabei sollte Höhe und Breite der View definiert und das ID Attribut gesetzt sein. Die zweite Variante erstellt das Inhalts Element dynamisch und kann über die ID *<id der Popover View>Content* angesprochen werden.

Als Parameter wird wieder ein assoziatives Arrays genutzt, dessen Elemente in der Abbildung 14: Parameter der Popover View erläutert werden:

Bezeichnung	Value	Beschreibung
id	String	Die ID des neuen HTML Elements
contentID	String	Die ID des vorher definierten HTML- div-Tags für den Inhalt. Dieser Parameter ist optional
arrowDir	„top“ „left“ „right“ „bottom“	Durch diesen Schlüssel wird definiert, wo der Pfeil an der View erscheinen soll. Dadurch wird auch die Lage der View auf dem Bildschirm definiert
offSet	Int	Durch diesen Parameter kann die View nach oben/unten beziehungsweise recht/links am Pfeil entlang verschoben werden. Ein Positiver Wert steht für die Pixel nach unten/rechts ein negativer in die jeweils andere Richtung.
width	Int	Dieser optionale Parameter definiert die Breite der View, wenn das Inhalts Element dynamisch erstellt wird.
height	Int	Dieser optionale Parameter definiert die Höhe der View, wenn das Inhalts Element dynamisch erstellt wird.
pointAt	Int	Als Parameter ist eine HTML- ID zu übergeben, auf das die Popover-View zeigen soll.

Abbildung 14: Parameter der Popover View

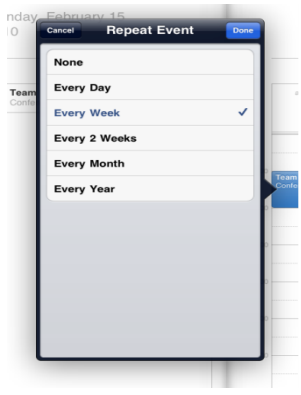

Abbildung 15: Die iOS Popover View (Apple Inc., 2011, S. 103)

Abbildung 16: Die neue Popover View

JavaScript-Code der Funktion createPopover:

```
function createPopover(probs){
   var popover = dojo.create('div',{'id':probs.id});
   popover.style.position='absolute';
   var arrow = dojo.create('div');
   arrow.style.width='1px';
   arrow.style.height='1px';
   arrow.style.position = "relative";
   arrow.style.zIndex=10;
   var pos = dojo.position(dojo.byId(probs.pointAtId));
   var content;
   if(probs.offSet==null){
      probs.offSet=0;
   }
   if(probs.contentId!=null && probs.contentId!=""){
      content=dojo.byId(probs.contentId);
   }
   else{
      content = dojo.create('div',{'id':probs.id+"Content"});
      probs.contentId=probs.id+"Content";
      content.style.height=probs.height+"px";
      content.style.width=probs.width+"px";
   }
   content.style.position='absolute';
   content.className="popover";
   popover.appendChild(arrow);
   popover.appendChild(content);
   dojo.body().appendChild(popover);
   var shadow;
   switch(probs.arrowDir){
      ...
      case 'bottom':
           arrow.className="popoverArrowBottom";
         popover.style.top=(pos.y-22+5) +"px";
         popover.style.left=(pos.x + pos.w/2+5)+"px";
         pos = dojo.position(content);
         content.style.top=(-pos.h-5)+'px';
         content.style.left=(-pos.w/2+22-5+probs.offSet)+"px";
         shadow = dojo.clone(arrow);
         shadow.className="popoverShadowBottom";
         arrow.appendChild(shadow);
         pos = dojo.position(shadow);
         shadow.style.top=-22-5+'px';
         shadow.style.left=-22-5+'px';
         break;
         ...
   }
   var cancel = dojo.create('div',{'ontouchend':function(evnt){
                        var pop = dojo.byId(probs.id);
                        pop.style.display='none';
                        }});
```

```
    cancel.style.position='absolute';
    cancel.style.backgroundColor='transparent';
    pos = dojo.position(dojo.body());
    console.log(pos);
    cancel.style.width= pos.w+"px";
    cancel.style.height=pos.h+"px";
    popover.appendChild(cancel);
    pos=dojo.position(cancel);
    cancel.style.left=-pos.x+"px";
    cancel.style.top=-pos.y+"px";
    cancel.style.zIndex=8;
    return popover;
}
```

2.3.3.3 Alert

“An alert pops up in the middle of the application screen and floats above its views [...] Users must dismiss the alert before they can continue using the currently running application [...]“ (Apple Inc., 2011, S. 116).

Für eine einfache Hinweis-Box kann anstelle einer Dojox-Funktion die PhoneGap-Funktion *navigator.notification.alert* mit den Parametern *message, alertCallback, [title], [buttonLabels]*) genutzt werden (vgl. Nitobi).

2.3.3.4 Action Sheet

„[...]On iPad, an action sheet is always displayed within a popover; it never has full-screen width“ (Apple Inc., 2011, S. 120).

Abbildung 17: Das iOS Action Sheet (Apple Inc., 2011, S. 120)

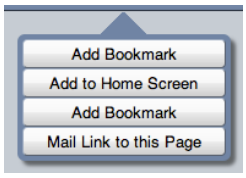

Abbildung 18: Das neue Action Sheet

Ein Action Sheet kann entsprechend dieser Definition durch eine Popover-View erstellt werden, die ausschließlich Schaltflächen enthält Abbildung 18: Das neue Action Sheet.

2.3.3.5 Modal View

„A modal view covers the entire application screen, which strengthens the user's perception of entering a separate, transient mode in which they can accomplish something“ (Apple Inc., 2011, S. 122).

Für die Dateneingabe ist die Modal View vorgesehen. Um eine Modal View zu erstellen, muss die JavaScript-Funktion *createModalView* ausgehführt werden. Dabei können auch innerhalb der Modal View verschiedene Templates angezeigt werden. Das bedeutet, dass nur eine Modal View pro Applikation erstellt werden muss. Optional können aber auch mehrere deklariert werden.

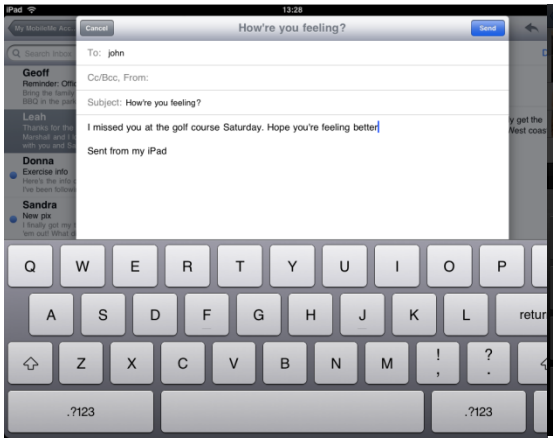

Abbildung 19: Die iOS Modal View (Apple Inc., 2011, S. 122)

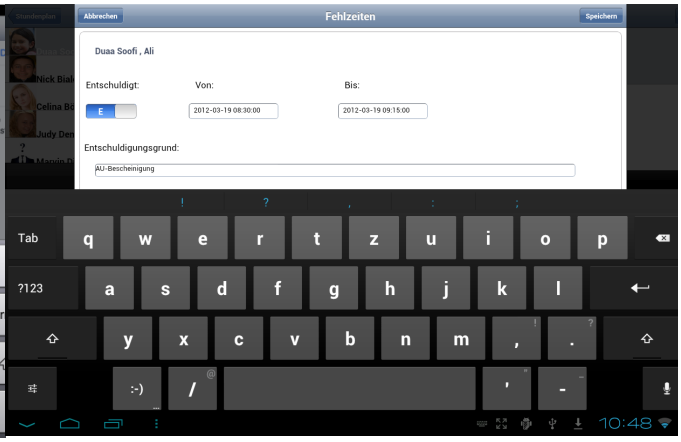

Abbildung 20: Die neue Modal View

Bezeichnung	Value	Beschreibung
id	String	Die ID des neuen HTML Elements (Modal View)
contentID	String	Ein optionaler Parameter, der eine ID eines HTML Elements enthält, welches als Inhalt in dem Modal View angezeigt werden soll
scrollable	Boolean	ein Parameter, der den Modalview Scrollable macht

Abbildung 21: Parameter der Modal View

JavaScript-Code der Funktion createModalView:

```
function createModalView (probs){
        var mod = dojo.create('div',{'id':probs.id});
        mod.className="modalView";
        mod.style.height=window.outerHeight+'px';
        var lBorder = dojo.create('div');
        lBorder.className="modalViewLBorder";
        var rBorder = dojo.create('div');
        rBorder.className="modalViewRBorder";
        mod.appendChild(lBorder);
        mod.appendChild(rBorder);
        dojo.body().insertBefore(mod,dojo.body().firstChild);
        mod.style.display="none";
        return mod;
}
```

2.3.3.6 Activity Indicator

„An activity indicator spins while a task is progressing and disappears when the task completes. Users do not interact with an activity indicator. By default, an activity indicator is white“ (Apple Inc., 2011, S. 124).

Der Activity Indicator kann durch eine CSS Klassendefinitionen visualisiert werden. Dabei benötigt man 12 HTML Div-Elemente, die in einem Container Div-Element eingebettet sind. Die CSS Definition von Jason Z. befindet sich in der Datei waitingIcon.css (Z., 2010).

Abbildung 22: Der iOS Activity Indicator (Apple Inc., 2011, S. 124)

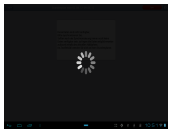

Abbildung 23: der neue Activity Indicator

2.3.3.7 Detail Disclosure Button

“Users tap a detail disclosure button to reveal additional information or functionality related to a specific item. The additional details or functionality are revealed in a separate view“ (Apple Inc., 2011, S. 126).

Der Detail Disclosure Button kann durch einen normalen HTML-Button dargestellt werden. Lediglich das *class*-Attribut muss als DetailDisclosureButton gesetzt sein, damit die Formatierung aus der *DetailDisclosureButton.css*, die durch die CSS *additionalUIElements.css* inkludiert wird, angewandt werden kann. In Abbildung 24: Der iOS Detail Disclosure Button wurde dieser mit einer Popover View kombiniert.

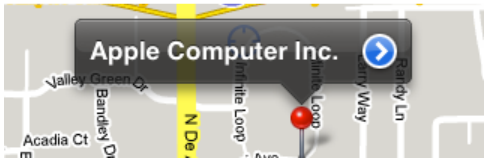

Abbildung 24: Der iOS Detail Disclosure Button (Apple Inc., 2011, S. 126)

Abbildung 25: Der neue Detail Disclosure Button

2.3.3.8 Info Button

“An Info button reveals configuration details about an application, often on the back of the current view“ (Apple Inc., 2011, S. 126).

Der Info-Button kann durch die Definition eines *dojox.mobile.ToolBarButton* Elementes erstellt werden. Dabei muss das Attribut *icon* mit der Pfadangabe zur Datei *infoBtn.png* gesetzt werden.

2.3.3.9 Search Bar

Die Eigenschaften einer Search Bar werden von Apple wie folgt definiert.

„A search bar looks like a text field with rounded ends. By default, a search bar displays the search icon on the left side. When the user taps a search bar, a keyboard appears; when the user is finished typing search terms, the input is handled in an application-specific way“ (Apple Inc., 2011, S. 131).

Um eine Search Bar zu erstellen, wird die JavaScript-Funktion *createSearchBar* aus der Datei additionalUIElements.js aufgerufen. Das Aussehen des Elements ist in der CSS-Datei Searchbar.css definiert, die wiederum in Datei additionalUIElements.css importiert wird.

Beim Aufruf der Funktion wird ein assoziatives Array als Übergabeparameter verwendet. Als Rückgabewert wird ein Container div-Element zurückgegeben, in dem die Search Bar enthalten ist. Zulässige Parameter sind:

Bezeichnung	Value	Beschreibung
id	String	Die neue Id, die das Textfeld der Search-Bar bekommen soll.
placeholder	String	Ein default Text, der in dem Textfeld angezeigt werden soll, wenn der optionale Parameter nicht gesetzt ist.
hasClearBtn	Boolean	Wenn dieser Parameter auf true gesetzt ist, wird der Button zum Löschen angezeigt. Kann nur gesetzt sein wenn *hasBookmarkBtn* nicht gesetzt ist.
hasBookmarkBtn	Boolean	Ist dieser Parameter auf true gesetzt ist, wird der Button für die gespeicherten Einträge angezeigt. Dieser Parameter kann nur gesetzt sein, wenn *hasClearBtn* nicht gesetzt ist.
parentId	String	Optionaler Parameter, welcher die ID des Eltern-Elements, in das die Search Bar eingefügt werden soll, enthält.

Abbildung 26: Parameter der Search Bar

JavaScript-Code der Funktion createModalView:

```
function createSearchBar(prob){
   var container = dojo.create('div',{'class':'searchBar'});
   var el = dojo.create('input',{'id':prob.id,'placeholder':prob.placeholder});
   dojo.byId(container).appendChild(el);
   if(prob.hasClearBtn){
      var clearBtn = dojo.create('div',{'class':'searchBarClearBtn'});
       dojo.byId(container).appendChild(clearBtn);
      clearBtn.ontouchend=function(){
            dojo.byId(prob.id).value="";

         }
   }
   if(prob.hasBookmarkBtn){
      var book = dojo.create('div',{'class':'searchBarBookmarkBtn'});
      dojo.byId(container).appendChild(book);
      book.ontouchend = prob.bookmarkFunction;
```

```
    }
    el.onkeydown=prob.keydownFunction;
    if(prob.parentId!=null){
        dojo.byId(prob.parentId).appendChild(container);

    }
    return container;

}
```

Abbildung 27: Die iOS Search Bar (Apple Inc., 2011, S. 131)

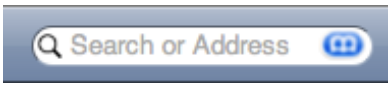

Abbildung 28: Die neue Search Bar

2.4 Zusammenfassung

Mit dem in diesem Kapitel diskutierten und beschriebenen Vorgehen ist es prinzipiell möglich, plattformunabhängige Applikationen zu erstellen, welche die Funktionsmerkmale von nativen Anwendungen erfüllen. Da das PhoneGap-Framework es nicht ermöglicht, auf Multi-Touch-Events bei der Gestensteuerung zu reagieren, sollte dieses Event-Handling als wichtiger Aspekt nativer Applikationen für Tablett-Systeme, durch JavaScript emuliert werden (vgl. Gundersen). Darüber hinaus kann, durch Abfrage des Betriebssystems, zur Laufzeit das Aussehen der Benutzungsoberfläche mit Hilfe von CSS an das jeweilige Zielsystem angepasst werden. Hierfür werden separate CSS-Dateien bereitgestellt.

3 Rahmenbedingungen für ein digitales Klassenbuch

Neben den oben beschrieben beschriebenen programmtechnischen Voraussetzungen für die Implementierung eines elektronischen Klassenbuchs werden im Folgenden weitere Rahmenbedingungen und notwendige Voraussetzungen diskutiert.

3.1 Der bildungspolitische Auftrag

Das Land Bremen hat im Jahr 2005 begonnen Maßnahmen zur Senkung der Wiederholerquote im Schulbereich der öffentlichen allgemeinbildenden Schulen durchzuführen.

Obwohl erste Erfolge nachweisbar sind, beginnen nach der internen Statistik der Senatorin für Bildung, Wissenschaft und Gesundheit (SfBWuG) immer noch rund 40% (2010/2011 805 Schüler) der bremischen Absolventen aus den 10. Jahrgangsstufen ihre berufliche Laufbahn im sogenannten Übergangssystem, um etwaige Defizite ihres Schulabschlusses zu kompensieren (Hansen, 2011). Hierbei werden als Übergangssysteme diejenigen Bildungsgänge bezeichnet, die nicht zu einem direkten Berufsabschluss oder zum Erwerb einer Hochschulzugangsberechtigung führen. Einen statistisch gesicherten Hinweis auf mögliche Ursachen lieferte die 2009 vorgelegte Dunkelfeldstudie mit der ausführlichen Darstellung des Schulschwänzens. Im Zusammenhang mit der Initiative *Stopp der Jugendgewalt*, die der bremische Senat auf den Weg brachte, wurde dieses Verhalten unter verschiedenen Blickwinkeln wissenschaftlich untersucht.

“Als Schulschwänzen wird das nicht legitimierte Fernbleiben vom Schulunterricht bezeichnet, dass auch einen Verstoß gegen schulgesetzliche Bestimmungen darstellt.

Davon abzugrenzen ist die Schulverweigerung. […] Ergebnisse britischer, amerikanischer und niederländischer Studien zeigen, dass das gelegentliche Schwänzen ein ‚normales’ Phänomen des Jugendalters ist. [...] Untersucht wurde [...] die Verbreitung des Schulschwänzens durch die Befragung von Lehrkräften und Jugendlichen. Die Lehrkräfte unterschätzen danach zwar insgesamt die Verbreitung

des Schulschwänzens systematisch. Hinsichtlich der hier besonders relevanten Quote der Jugendlichen, die besonders intensiv schwänzen, sind ihre Wahrnehmungen aber recht zutreffend. Ihre Einschätzungen sind sehr nahe an den Befunden auf der Basis von Selbstberichten […]. Auffallend ist allerdings, dass immer noch ein beträchtlicher Anteil der gehäuft schwänzenden Jugendlichen bislang ohne Konsequenzen geblieben ist [...] (Wetzels & Brettfeld, 2009)“.

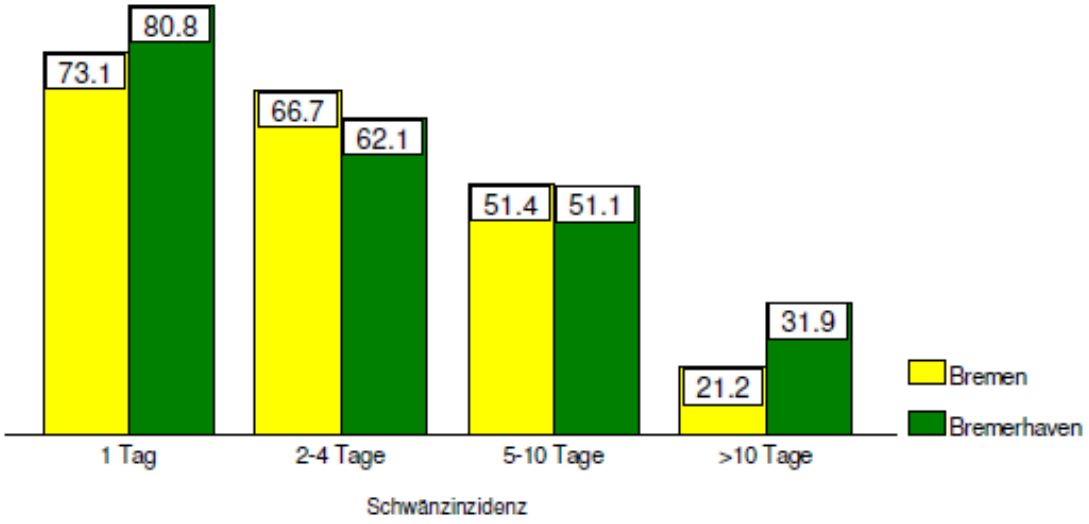

Abbildung 29: Rate der Schwänzer, die keine Reaktion auf ihr Schwänzen erlebt haben nach Intensität des Schulschwänzens (Wetzels & Brettfeld, 2009, S. 102)

Darüber hinaus gilt die Korrelation zwischen Schulabsentismus und keinem oder schlechtem schulischen Abschluss als statistisch gesichert.

Das bisherige, im Wesentlichen auf dem traditionellen Klassenbuch basierende Verfahren zur Dokumentation von Fehlzeiten zeichnet sich zum einen durch zahlreiche Medienbrüche bei der Erfassung und Übertragung der Daten in die Schülerakten aus und hat zum anderen den Nachteil, dass eine zeitnahe Auswertung zu diagnostischen Zwecken durch den Klassenbetreuer oder Klassenbetreuerin bzw. die Schulleitung nur mit unverhältnismäßig großem Aufwand möglich ist.

Die Senatorin für Bildung, Wissenschaft und Gesundheit, Renate Jürgens-Pieper, hat deshalb Ende 2010 den Auftrag zu einem Pilotvorhaben mit dem Arbeitstitel

elektronisches Klassenbuch (EKB) erteilt, in dem eine geeignete Softwarelösung für ein digitales Klassenbuch zur elektronischen Erfassung von Fehlzeiten passend zum Schulverwaltungsprogramm der Bremer Schulen definiert werden soll (vgl. Hansen, 2011).

Die erforderlichen finanziellen Mittel, um an jedem Oberschul- und Gymnasialstandort eine Infrastruktur zu erstellen, die in jedem Unterrichtsraum die elektronische Erfassung von Unterrichtsdaten ermöglicht, ist im Haushalt 2012 der Freien Hansestadt Bremen im Sonderprogramm *Umbau der Verwaltung und Infrastruktur* beantragt worden (vgl. Hansen, 2011).

Durch eine elektronische Fehlzeitenerfassung im Zusammenspiel mit der Einführung einer elektronischen Schülerakte soll eine Reduktion des Arbeitsaufwandes von Schulleitungen erzielt werden. Die Einbettung in das Schulverwaltungsprogramm führt zu einem systematischen Reporting, welches die Informationsflüsse beschleunigt und gleichzeitig die bestehende Sicherheitsinfrastruktur nutzt.

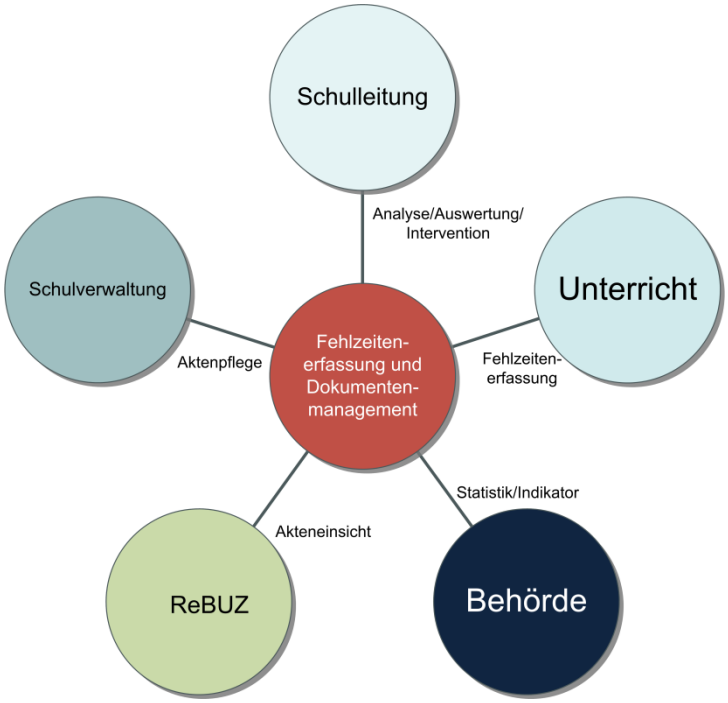

Abbildung 30: Institutionen die an der Erhebung und Auswertung von Daten des digitalen Klassenbuches beteiligt sind

Zur automatischen Integration von Fehlzeiten und Sozialverhalten in die Schulverwaltungssoftware ist der Aufbau einer flächendeckenden und bedarfsgerechten Infrastruktur in den Unterrichtsräumen zwingend erforderlich. Die hierbei auftretenden datenschutzrechtlichen und sicherheitstechnischen Anforderungen gehen über die bestehenden Regularien der Datenverarbeitung in der Schulverwaltung hinaus Bremisches Schuldatenschutzgesetz (BremSchulDSG) und müssen vor dem eigentlichen Verfahrensbeginn vom Schulträger mit einer angemessenen Sicherheitsarchitektur und Novellierung des Erlasses Nr. 03/2012 „Richtlinien über die Verarbeitung von personenbezogenen Daten und zur Führung von Schullaufbahnakten in der Stadtgemeinde Bremen“ begegnet werden (vgl. (Beck-online die Wissensdatenbank, 2007); (Hansen, 2011); (SfBWuG, 2012)).

Neben der bildungspolitischen Dimension des Vorhabens ergibt sich als Nebeneffekt ein Einsparpotential aus der Verminderung des Verwaltungsaufwandes und der Transaktionskosten zur Führung und Auswertung von Schülerakten gegenüber der bisherigen Form und wirkt sich damit positiv insbesondere auf die Arbeit der Schulleitungen aus. Durch die systematische Erfassung von Fehlzeiten und Sozialverhalten wird eine rechtzeitige Diagnostik durch die Schulleitung und somit auch eine gezielte Intervention durch die Regionalen Unterstützungszentren (ReBUZ) ermöglicht (vgl. Hansen, 2011).

3.2 Die Ausgangssituation

Für den Aufbau einer bedarfsgerechten Infrastruktur in Verbindung mit einer geeigneten Softwarelösung für die Erfassung von Fehlzeiten und Sozialverhalten sind, gerade unter knappen Ressourcenbedingungen, die vorhandenen schulorganisatorischen und technischen Rahmenbedingungen für die zu treffenden Technisch-Organisatorischen Maßnahmen von entscheidender Bedeutung. Daher werden diese Rahmenbedingungen im Folgenden besonders hervorgehoben.

3.2.1 Die schulorganisatorische Rahmenbedingungen

Im Rahmen der *Neuordnung der Aufgabenwahrnehmung in der freien Hansestadt Bremen* wurde bereits 2001 ein Teilprojekt zur Softwareauswahl für eine Schulverwaltungssoftware in Zusammenarbeit mit der Unternehmensberatung Roland Berger durchgeführt.

An Bremer Schulen existierte mit Ausnahme der beruflichen Schulen bis dahin keine einheitliche Schulverwaltungssoftware. Die allgemeinbildenden Schulen nutzten verschiedene EDV-gestützte Verfahren, zum Beispiel EXCEL, oder Eigenprogrammierungen mit ACCESS. Einzelne Schulen hatten auf eigene Initiative unterschiedliche Schulverwaltungssoftware installiert. In den Schulen wurden die Schülerdaten in Schülerakten und auf Karteikarten geführt. Der hierdurch entstehende hohe manuelle Aufwand führte zu ineffizienter Aufgabenwahrnehmung, sowohl in der Schule als auch in der Behörde. Eines von mehreren Problemen lag in der Doppelerfassung für die Erstellung der jährlichen Statistiken. Im Laufe der Untersuchung wurden über 90 Softwareprodukte mit unterschiedlichem Funktionsumfang identifiziert.

Entscheidend für die Auswahl von Magellan der Firma Stüber Software GMBH als Schulverwaltungssoftware war das Client/Server-Konzept mit dezentral/zentraler Datenhaltung. Darüber hinaus war zum damaligen Zeitpunkt diese Software die einzige, die über eine moderne relationale und replikationsfähige Datenbank verfügte. Ergänzt wurde die Ausstattung durch zwei alternative Stundenplansoftwarepakete da Vinci, ebenfalls von der Firma Stüber und Untis von Gruber und Petters (vgl.SfBW, 2002).

Nachdem 2002 die erste Novellierung des Bremischen Gesetzes zum Datenschutz im Schulwesen die Verarbeitung von Schülerdaten in elektronischer Form absicherte, konnte 2003 der flächendeckende Echtbetrieb aufgenommen werden.

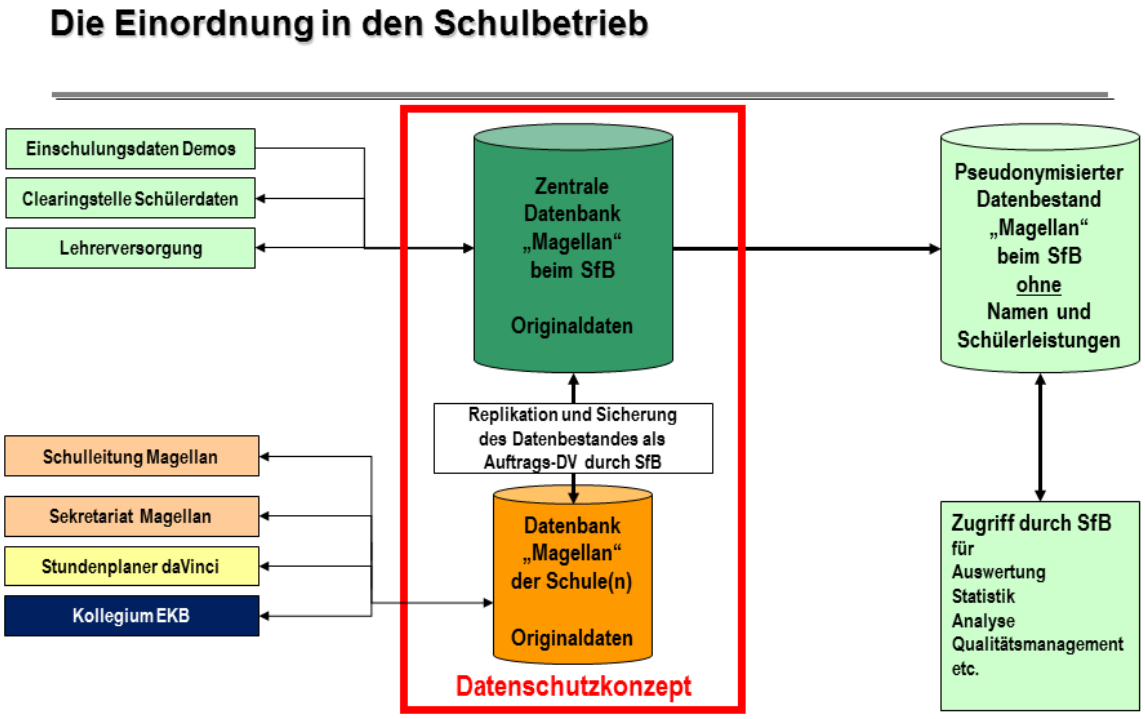

Abbildung 31: Einordnung der elektronischen Fehlzeitenerfassung (EKB) in den Schulbetrieb

Ein elektronisches Klassenbuch lässt sich in diese Client/Serverstruktur technisch gesehen nahtlos integrieren, stellt aber durch die dezentrale Erfassung von Daten außerhalb der eigentlichen Kernverwaltung (Schulleitung und Schulsekretariat) durch Lehrerinnen und Lehrer hinsichtlich der Datensicherheit und des Datenschutzes neue Anforderungen.

3.2.1.1 Das aktuelle Klassenbuch

In der derzeitigen traditionellen Form des Klassenbuches werden folgende Informationen in Buchform handschriftlich für jede Klasse separat festgehalten:

- Unterrichtsinhalt
- Fehlzeiten
- Fehlverhalten
- Bemerkungen zum Sozialverhalten
- Klassenarbeiten
- Entschuldigungen

- Hausaufgaben
- Klassendienste
- Geplante Aktivitäten
- Kontaktinformationen (Eltern/Sorgeberechtigte)
- Liste der Lehrkräfte
- Schüler-/Klassen-/Elternsprecher
- Stundenplan
- Gruppenzuordnung (z.B. AGs)
- Bücherliste
- Ausbildungsbetriebe (an beruflichen Schulen)
- Vertretungsinformationen

3.2.1.2 Anforderungen an das elektronische Klassenbuch

Für die konkrete Umsetzung einer Applikation für den Schulbetrieb gibt es neben den allgemeinen Anforderungen an eine Realisierung für Tablettsysteme noch weitere spezifische Merkmale, die für eine erfolgreiche Einführung erfüllt werden müssen.

Im Auftrag der Senatorin für Bildung Wissenschaft und Gesundheit führte das Institut für Informationsmanagement Bremen GmbH (ifib) 2011 eine Umfrage an ausgewählten Schulen durch und erstellte daraus ein Anforderungsprofil für ein elektronisches Klassenbuch. Die folgende Tabelle listet diese auf. Für die Antworten wurden den Schulen keine Vorgaben gemacht, sondern die Schulen konnten frei antworten.

1. Das System muss den Unterrichtsinhalt festhalten.
2. Der Zugriff zum elektronischen Klassenbuch von zu Hause aus und von privaten Geräten muss möglich sein.
3. Das System muss eine detaillierte Übersicht über die Fehlzeiten der Schülerinnen und Schüler anbieten.
4. Das System muss schnell zu bedienen sein, Einträge dürfen nicht viel länger dauern als im herkömmlichen Klassenbuch.

5. Das System muss automatisierte Benachrichtigungen verschicken (zum Beispiel bei fehlenden Schülerinnen und Schülern oder fehlenden Einträgen von Stunden).
6. Mobile Endgeräte für die Eingabe in das elektronische Klassenbuch sind eher gewünscht als feste Rechner.
7. Das System muss klar verständlich und einfach zu benutzen sein.
8. Das System muss die gesammelten Daten gut verwendbar aufbereiten (zum Beispiel für die Verwendung am Elternsprechtag).
9. Das System muss eine Notenfunktionalität haben.
10. Das System muss detaillierte Informationen zum Unterrichtsinhalt festhalten können.
11. Das System muss von einem Smartphone aus benutzbar sein.
12. Das System muss den Vertretungsplan integrieren.
13. Das System muss den Stundenplan integrieren.
14. Das System muss ähnlich wie das herkömmliche Klassenbuch aufgebaut sein.
15. Fremdzugang zu Daten muss verhindert werden.
16. Das System muss mit Lernplattformen kombinierbar sein.
17. Das System muss die Nachbereitung von Stunden erlauben.
18. Das System muss die Vorbereitung von Stunden erlauben.
19. Das System muss Fehlverhalten von Schülerinnen und Schülern festhalten.
20. Das System muss schulinterne Formulare ersetzen.
21. Das System muss barrierefrei sein.
22. Das System muss Verspätungen aufnehmen können.
23. Das System muss mit MyFuNe kombinierbar sein.
24. Das System muss Vertretungsmaterial aufnehmen können.
25. Das System muss Serienbriefe verschicken können.

26. Das System muss zur Planung von Ereignissen verwendbar sein.

Abbildung 32: Anforderungsspezifikation der Schulen (ifib, 2011, S. 2)

Auf direkte Nachfrage bezüglich Punkt 11 stellte sich heraus, dass hier nicht nur Smartphones, sondern mobile Geräte im Allgemeinen gemeint waren (Hansen, 2011).

3.2.2 Die unterrichtsorganisatorischen Rahmenbedingungen

„Die Anwesenheitslisten aus dem Klassenbuch werden an den meisten Schulen halbjährlich überprüft." So lautet die Auskunft von Herrn Hövelmann, Leiter des „Betriebs der integrierten, einheitlichen Schulverwaltungssoftware Magellan und daVinci (vgl. Hövelmann, 2011). Laut einer Befragung des ifib wird das Fehlen von Schülern nach Lehreraussagen von Schule zu Schule unterschiedlich schnell bemerkt.

„Nur vereinzelt wurde erwähnt, dass das Fehlen eines Schülers immer sofort bemerkt würde. Viele der Befragten schilderten es als unregelmäßig oder lehrerabhängig und auch abhängig vom Alter der Schüler. In den unteren Klassenstufen wird ein Fehlen eher bemerkt und auch eher agiert" (ifib, 2011).

Auch für Benachrichtigung der Eltern über das Fehlen ihrer Kinder im Unterricht gibt es keine verbindliche Vorgaben und somit keine einheitliche Handhabung in den vom ifib befragten Schulen. So werden in einigen Schulen bereits bei jedem bemerkten Fehlen sofort die Eltern informiert, oft aber geschieht dies auch erst am dritten Tag, manchmal nur sporadisch und unregelmäßig. Damit decken sich diese Ergebnisse auch noch 2012 mit der 2009 durchgeführten Dunkelfeldstudie.

3.2.3 Die bildungspolitischen Rahmenbedingungen (Inklusive Beschulung)

„Das Bremer Schulgesetz von 2009 formuliert in § 3 Absatz 4 als erstes Schulgesetz in Deutschland den Auftrag, dass sich alle Schulen zu inklusiven Schulen entwickeln sollen" (Götz, 2011).

„Bremische Schulen haben den Auftrag, sich zu inklusiven Schulen zu entwickeln. Sie sollen im Rahmen ihres Erziehungs- und Bildungsauftrages die Inklusion aller Schülerinnen und Schüler unabhängig von ihrer ethnischen Herkunft, ihrer

Staatsbürgerschaft, Religion oder einer Beeinträchtigung in das gesellschaftliche Leben und die schulische Gemeinschaft befördern und Ausgrenzungen einzelner vermeiden“ (Beck-online Die Wissensdatenbank, 2009).

„Das heißt, Schülerinnen und Schüler in all ihrer Verschiedenheit lernen gemeinsam, wenn auch nicht immer mit den gleichen Zielen.

Ausdrücklich soll der Unterricht und das weitere Schulleben für behinderte und nichtbehinderte Schülerinnen und Schüler in allen Schulstufen gemeinsam gestaltet werden.“ (Götz, 2011).

Diese gesetzlichen Auflagen stellen eine besondere Herausforderung für das Schulsystem dar.

Zur Bewältigung dieser Aufgabe sieht das Schulgesetz zwei abgestufte Unterstützungssysteme vor. Im Zentrum für unterstützende Pädagogik (ZuP-Team) arbeiten Lehrkräfte mit sonderpädagogischer Ausbildung, sowie andere Lehrkräfte mit besonderem Förderauftrag für Sprachförderung oder zur Kompensation von Lese-Rechtschreibschwächen. Ergänzt wird dieses schulbezogene Angebot durch die Regionalen Unterstützungszentren (ReBUZ).

„Ein ReBUZ bietet umfängliche Einzelfallberatung sowie Diagnostik an. Diese betrifft insbesondere den sozial-emotionalen Bereich sowie Legasthenie, Dyskalkulie aber auch Hochbegabung. Das Zentrum koordiniert und unterstützt außerdem Konzepte und Maßnahmen zur Gewaltprävention an Schulen. Temporäre Bildung, Erziehung und Betreuung von Schülerinnen und Schülern, deren Lern- und Sozialverhalten eine Beschulung in der allgemeinen Schule nicht zulässt, soll in einem ReBUZ in kleinen Gruppen und in Praktika erfolgen. Ziel ist die Wiedereingliederung in die allgemeine Schule. Außerdem gehören Schullaufbahnberatung, Schulvermeidungsprogramme und Kooperation mit allen in Frage kommenden Institutionen aus den Bereichen Soziales, Jugend, Inneres und Justiz zu den Aufgaben eines regionalen Unterstützungs- und Beratungszentrums (Götz, 2011).“

Um ihre Aufgaben angemessen wahrnehmen zu können, ist es erforderlich, dass den Unterstützungssystemen zeitnah alle Vorgänge, die zum Verständnis des

Schülers und der jeweiligen individuellen Situation beitragen, zugänglich gemacht werden. Zum Lern- und Sozialverhalten gehören insbesondere Informationen:

- Nichterfüllung der Schulpflicht
- Ordnungsmaßnahmen
- Einzelunterricht
- Längerfristige Beurlaubungen
- Freistellung vom Unterricht in einzelnen Fächern
- Verhaltensdaten
- Atteste und schulärztliche Vorgänge
- Behinderungen und chronische Krankheiten
- Daten über besondere therapeutische Maßnahmen und deren Ergebnisse
- Den Schüler betreffender Schriftverkehr und Gesprächsnotizen

Ordnungsmaßnahmen sind im Bremer Schulblatt in der *Verordnung über das Verfahren beim Erlass von Ordnungsmaßnahmen in der Schule* definiert (SfBW, 1998, §§1-25).

Auffällig ist, dass derzeit mit den Schlüsseltabellen zum Lern- und Sozialverhalten nur negativ besetzte Merkmale erfasst werden können.

3.2.4 Die rechtlichen Rahmenbedingungen (Datenschutz)

Die rechtlichen Rahmenbedingungen für die Verarbeitung personenbezogener Daten sind im Bremischen Schuldatenschutzgesetz beschrieben (vgl. Beck-online die Wissensdatenbank, 2007).

Dieses gliedert sich in drei Teile:

- Teil 1 Datenverarbeitung in den Schulen
- Teil 2 Datenverarbeitung beim Senator für Bildung und Wissenschaft und beim Magistrat Bremerhaven
- Teil 3 Datenverarbeitung beim Schulärztlichen und Schulpsychologischen Dienst

Für den Einsatz auf Geräten außerhalb der Schulverwaltung ist die Änderung im ersten Teil des §3. Abs. 2 des Bremischen Schuldatenschutzgesetzes von

besonderer Bedeutung. Diese am 10.03.2007 in Kraft getretene Novellierung eröffnet einen rechtssicheren Weg auch sensible personenbezogene Daten auf privaten (mobilen) Endgeräten zu verarbeiten.

„1 Lehr- und Betreuungskräfte, die sich schriftlich zur Beachtung der datenschutzrechtlichen Vorschriften verpflichtet und sich mit der Überwachung durch den behördlichen Datenschutzbeauftragten und den Landesbeauftragten für den Datenschutz einverstanden erklärt haben, dürfen zur Erfüllung ihrer Aufgaben private Datenverarbeitungsgeräte zur Verarbeitung personenbezogener Daten von Schülerinnen und Schülern verwenden.

2 Sie haben sicherzustellen, dass diese Daten vor dem Zugriff Dritter geschützt sind und spätestens nach dem Ende des jeweils nächsten Schuljahres gelöscht werden“ (Beck-online die Wissensdatenbank, 2007).

Dieses Gesetz wird im Erlass Nr. 03/2012 Richtlinien über die Verarbeitung von personenbezogenen Daten und zur Führung von Schullaufbahnakten in der Stadtgemeinde Bremen vom 29.02.2012 konkretisiert und bildet den technisch-organisatorischen Hintergrund für den weiter unten beschriebenen Lösungsansatz für eine dezentrale Erfassung der Fehlzeiten und des Sozialverhaltens auf mobilen Endgeräten. Die derzeitige Richtlinie bezieht sich ausdrücklich in Kapitel 2.2 Datenschutz auf nicht automatisierte Dateien:

„In der SchLA (Schullaufbahnakte) werden Dokumente mit personenbezogenen Daten abgelegt. Sie stellen eine nicht automatisierte Datei im Sinne des § 2 Abs. 3 Nr. 4 BremDSG dar (Datenverarbeitung).

Besonders sind die §§ 3 (Zugang und Nutzung der Daten), 4 (Einwilligung in die Datenverarbeitung, Unterrichtungspflicht der Schule), 7 bis 10 (Datenübermittlung), 19 (Aufbewahrung und Löschung) und 20 (Einsichts- und Auskunftsrecht) BremSchulDSG zu beachten“ (SfBWuG, 2012).

Die Übertragung dieser für die herkömmliche Aktenführung geltenden Regelungen auf eine digitale Aktenführung ist von der SfBWuG vor der flächendeckenden Einführung einer digitalen Lösung noch zu leisten.

3.2.5 Die technischen Rahmenbedingungen

Für die erfolgreiche Implementierung des Klassenbuches für den schulischen Alltag, spielt die vorhandene technische Ausstattung und die Netzinfrastruktur eine entscheidende Rolle.

3.2.5.1 Netzkonzept

Durch umfangreiche Sonderprogramme konnte ab 2001 in den Bremer Schulen schrittweise eine leistungsfähige flächendeckende *Wide Area Network* (WAN) *Infrastruktur* aufgebaut werden. Jeder der insgesamt 150 Schulstandorte ist mit einer Bandbreite von 2MBit/s bis 100MBit/s an das geschlossene Intranet der SfBWuG angebunden.

Die Einordnung in die Netzinfrastruktur

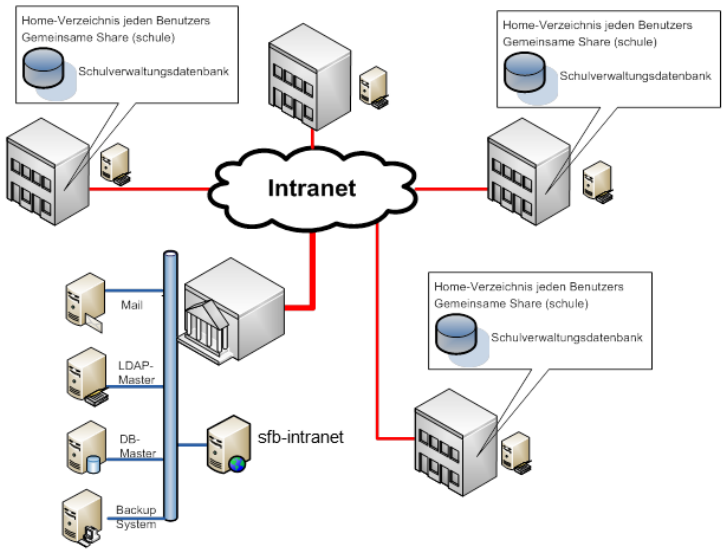

Abbildung 33: Die Netzinfrastruktur des Schulnetzes

Am Standort Rembertiring stehen die zentralen Serversysteme, welche tagesaktuell ihre Daten mit den jeweiligen Schulstandorten durch Replikationen abgleichen. In den Schulstandorten ist eine strukturierte Verkabelung mit durchgängig 100MBit vorhanden. Hier wird zwischen den edukativen Netz und dem Verwaltungsnetz unterschieden. Beide Netze sind logisch vollständig voneinander getrennt. An den

Schulstandorten wird die stationäre Verkabelung durch WLAN-Lösungen im Bereich des edukativen Netzes ergänzt.

3.2.5.2 Schulische Ausstattung

In fast allen Unterrichtsräumen ist mindestens ein Netzwerkanschluss an das edukative Netzwerk vorhanden. 2006 war darüber hinaus etwa die Hälfte aller vorhandenen Unterrichtsräume in den Schulen mit Rechnern ausgestattet.

„Die Anzahl an Unterrichtsräumen mit Medienausstattung ist damit sehr hoch. Die meisten Räume mit Rechnern haben die Schulen der Primar- und der Sekundarstufe II (über 70% der Räume sind mit Rechnern ausgestattet) sowie die Förderzentren (83%). Die wenigsten Räume mit Rechnern haben die Gymnasien und die Schulen der Sekundarstufe I (22% bzw. 24%). Im Schnitt hat eine Schule drei bis vier PC-Räume, 12 Klassenräume und drei Fachräume mit Computern, wobei sich diese Werte zwischen den einzelnen Schulformen stark unterschieden“ (Breiter, 2006).

In den Klassenräumen des Sekundarbereiches I wurde der Rechnerbestand zwar modernisiert, aber nicht wesentlich erhöht. Damit ist für die Realisierung einer flächendeckenden Erfassung von Fehlzeiten und Sozialverhalten zwar eine gute Netz-Infrastruktur vorhanden, jedoch ist die erforderliche Hardware in der Regel nur in Ausnahmefällen im Klassenraum in der notwendigen Qualität für Verwaltungsvorgänge vorhanden und wird auf Grund der begrenzten Ressourcen auch nur schrittweise beschafft werden können (vgl. Hansen, 2011).

In diesem Zusammenhang stellt die Ergänzung der Netzinfrastruktur der Schulen durch WLAN-Komponenten für die Nutzung von mobilen Geräten im Unterricht einen bedeutsamen Meilenstein in der Weiterentwicklung der unterrichtlichen Nutzung der Informationstechnik dar. Neben des noch zu realisierenden Mobile-Device-Managements (Apple Inc., 2011), (Symantec) bei der SfBWuG bildet dieses WPA2-verschlüsselte WLAN dann die notwendige Infrastruktur für die Nutzung eines digitalen Klassenbuches.

3.2.5.3 Sicherheitskonzept

Grundlage für das geplante Mobile-Device-Management bei der SfBWuG bildet *das Microsoft Exchange ActiveSync* (EAS). EAS ist ein auf *Hypertext Transfer Protocol* (http) und *Extensible Markup Language* (Xml) basierendes Protokoll, welches neben dem Austausch von Email, Kalenderinformationen, Kontakten, Aufgaben- und Adresslisten eine grundlegende Rechtesteuerung zentral bereitstellt. Damit ist es Administratoren unter anderen möglich, die Verschlüsselung eines Endgerätes zu erzwingen, oder den Zugang zum mobilen Endgerät mit einer Passworteingabe abzusichern. Der Komplexitätsgrad dieses Passwortes kann vom Administrator vorgegeben werden. Darüber hinaus ist es möglich bei mehrfacher Falscheingabe des Passwortes eine automatische Löschung der Daten auf dem Endgerät anzustoßen.

Es versteht sich von selbst, dass bei der Nutzung im Rahmen des digitalen Klassenbuches nur Geräte zum Einsatz kommen dürfen, die eine Umgehung des Sandboxkonzeptes ausschließen. Dies bedeutet, dass ein Root Zugang, wie er zum Beispiel beim iOS durch einen Jailbreak erfolgen kann, von dem Mobile-Device-Management erkannt und gemeldet werden muss.

Zusätzlich zu dem oben genannten Mobile-Device-Management, ist innerhalb der Applikation eine Authentifizierung des Nutzers vorgesehen.

Darüberhinaus erfolgt auch der Zugang zum Schulnetzwerk über eine *Virtual Private Network* (VPN)-Verbindung erst nach einer weiteren Authentifikation des Nutzers.

Bei der Übertragung von Daten über das Netzwerk greift die Standartverschlüsselung WPA2. Durch den Einsatz einer zertifikatsbasierten Verschlüsselung könnte die Sicherheit des Datentransfers zusätzlich zur WPA2 Verschlüsselung erhöht werden. Allerdings wird momentan innerhalb des Frameworks PhoneGap für iOS die Nutzung selbstsignierter Zertifikate nicht unterstützt. Aus Gründen der Plattformunabhängigkeit wurde daher in dieser Arbeit auf eine solche zusätzliche Verschlüsselung des Datentransfers verzichtet.

Neben diesen technischen Sicherheitsmaßnahmen erfolgt eine Protokollierung aller Modifikationen von Datensätzen auf Datenbankebene. Bei Erstellung eines

Datensatzes wird sowohl Uhrzeit und Datum sowie der Name des Erstellers im Datensatz gespeichert. Jede Veränderung des Datensatzes wird zusätzlich mit Datum und Uhrzeit sowie dem Namen des jeweiligen Nutzers separat protokolliert.

3.2.5.4 Datenbankbeschreibung

Die Daten aus dem schulorganisatorischen Alltag, welche durch Magellan oder die Stundenplansoftware genutzt und erhoben werden, sind in einer zentralen Datenbank gespeichert. Die Datengrundlage von Magellan in der Version 6 ist die Open Source-Datenbank Firebird. Es handelt sich um eine relationale Client/Server-Datenbank, auf die unter anderem per *Open Database Connectivity* (ODBC) direkt zugegriffen werden kann.

Bei der SfBWuG laufen sowohl die zentrale Datenbank als auch die dezentralen Datenbanken der Schulen unter Linux (Firebird auf Ubuntu 10.4 LTS). In PHP steht eine entsprechende Bibliothek zur Verfügung welche für die Entwicklung der Datenschnittstelle benutzt wurde.

Die aktuelle Magellan-Datenbank besteht in der Grundversion aus 214 Tabellen, 41 Views, sowie zahlreichen weiteren Datenbankobjekten (unter anderem Trigger, Prozeduren). Sie wurde während der fast zehn-jährigen Nutzung von der SfBWuG um mehr als 60 eigene Tabellen und Views erweitert. Durch den Zugriff auf diese Datenbank konnten die Magellan-Daten für die vorliegende Arbeit nutzbar gemacht werden.

3.3 Zusammenfassung

Der Einsatz eines digitalen Klassenbuches erscheint unter den gegebenen technischen Rahmenbedingungen realisierbar, allerdings sind vor einer flächendeckenden Einführung noch erhebliche organisatorische Hindernisse zu beseitigen. Zu diesen gehört die Anpassung des Erlasses Nr. 03/2012 *Richtlinien über die Verarbeitung von personenbezogenen Daten und zur Führung von Schullaufbahnakten in der Stadtgemeinde Bremen*, um eine Verarbeitung in Form automatisierter Dateien zu ermöglichen. Die zu lösenden datenschutz- und datensicherheitstechnischen Fragestellungen, die Erstellung eines

Datenschutzkonzeptes und die Beteiligung der Interessenvertretungen sind hierbei zu erwähnen.

Ein bislang von der SfBWuG durchgeführtes Pilotprojekt an zwei Schulen beschränkt sich auf die zwei am Markt verfügbaren Ergänzungen zum Schulverwaltungsprogramm, welche allerdings die Anforderungen des ifib nur unvollständig abbilden (Hansen, 2011). Vor diesem organisatorischen Hintergrund erfolgt, mit Zustimmung und Unterstützung der SfBWuG, die Entwicklung eines Prototyps für ein digitales Klassenbuch.

4 Umsetzung eines digitalen Klassenbuches

Vor der programmtechnischen Umsetzung des Klassenbuchs wurden die technisch-organisatorischen Rahmenbedingungen analysiert und unter Berücksichtigung der Befragung der Schulen zum Anforderungsprofil ein Umsetzungskonzept entwickelt (vgl. ifib, 2011).

4.1 Funktionsspezifikation

Nach eingehender Analyse der in Kapitel 3 aufgeführten Anforderungen in Abbildung 32: Anforderungsspezifikation der Schulen an ein digitales Klassenbuch kann folgende Zusammenfassung der Anforderungen für die Implementierung eines Prototyps festgelegt werden:

- Zusätzlich zu den im Rahmen des *Mobile-Device-Management* durchgeführten Sicherungs- und Zugangssteuerungen und der Verschlüsselung des WLANs während der Synchronisation, sollen Daten des Klassenbuches nur nach Authentifizierung des Nutzers angezeigt werden. Hierbei werden nur die Daten zur Verfügung gestellt die nach dem Rollenkonzept des Schulverwaltungsprogramms zugänglich sind.
- Es soll eine standardisierte Möglichkeit zur schnellen Erfassung von Fehlzeiten geben, welche gegebenenfalls durch Anmerkungen im Freitextformat, sowohl hinsichtlich des Fehlgrundes, als auch des Entschuldigungsgrundes modifiziert und kommentiert werden können. Für die Kurzeingabe sind innerhalb des Schulverwaltungsprogramms festgelegte Schlüssel bereitzustellen.
- Es soll eine standardisierte Möglichkeit zur Eingabe von Vermerken zum Schülerverhalten, dem sogenannten Lern- und Sozialverhalten, geben. Diese können durch Freitextfelder hinsichtlich des Anlasses und gegebenenfalls auch der getroffenen Maßnahme ergänzt und kommentiert werden. Für die Kategorisierung sind innerhalb des Schulverwaltungsprogramms definierte Schlüssel zu benutzen.
- Zur Unterstützung der Unterrichtsdokumentation und -planung soll für die Nutzer die Eingabe von Informationen zum Stundenverlauf, wie Stoffverteilung

und Hausaufgaben, in Form von Freitext aber auch in Form von Schlüsselwerten, möglich sein. Hierfür können innerhalb des Schulverwaltungsprogramms schuleigene Schlüssel festgelegt werden.

- Es muss eine intuitive Möglichkeit für die Nutzer vorhanden sein, um zwischen den verschiedenen Unterrichtsstunden zu navigieren.
- Es müssen Daten aus der Unterrichtssituation auch ohne Netzzugang erfasst werden können. Darüber hinaus muss die Applikation die Möglichkeit bieten, die Daten, insbesondere zum Stundenverlauf, auch außerhalb des Schulnetzes bearbeiten zu können.

Um den Dokumentationsumfang im Falle von Vertretungsunterricht bereitstellen zu können, ist ein performanter Abgleich zwischen Schulserver und mobilen Gerät eine wesentliche Voraussetzung.

4.2 Datenverwaltung

Im Rahmen des Schulverwaltungsprogrammes Magellan liegt eine Tabellenstruktur für ein elektronisches Klassenbuch vor. Die Bezeichnungen der relevanten Tabellen wurde aus Gründen der Konsistenz des Datenbankkonzeptes übernommen, obwohl einige Tabellennamen aber insbesondere Feldbezeichnungen wenig aussagekräftig und wenig systematisch sind.

4.2.1 Konzeption der Datenbank

Prinzipiell lassen sich Daten lokal mit Hilfe von PhoneGap sowohl als Files als auch in SQLite-Tabellen speichern. SQLite steht auf allen von PhoneGap unterstützten Systemen zur Verfügung beziehungsweise wird durch eine *World Wide Web Consortium* (W3C) Web SQL Database konforme Implementierung bereitgestellt (Nitobi). Für die Parametrisierung des Datenaustausches zwischen Schulserver und mobilen Endgeräten dient die Tabelle OPTIONS. Zurzeit sind nur die Schulnummer oder die Information, ob mit Schülerbildern synchronisiert werden soll, hinterlegt.

Eine LOGIN-Tabelle garantiert den gesicherten Zugriff auf die Daten außerhalb des Schulnetzes. Darin werden der Login des Nutzers, sowie das benutzte Passwort in verschlüsselter Form gespeichert. So können mehrere Personen das gleiche mobile Gerät nutzen, während zugleich sichergestellt ist, dass auf Grund der

Authentifizierung jedem Nutzer nur solche Daten angezeigt werden, für die eine Autorisierung vorhanden ist.

Zur Darstellung des Stundenplans Abbildung 48: Stundenplan Screen auf einem Android System dient die Tabelle STDRASTER, in der das Stundenraster der Schule mit Datum und Uhrzeit aggregiert wird. Das Feld *StundeVon* enthält die Angabe, um die wievielte Stunde es sich handelt, während in StdVon und StdBis das Zeitintervall enthalten ist.

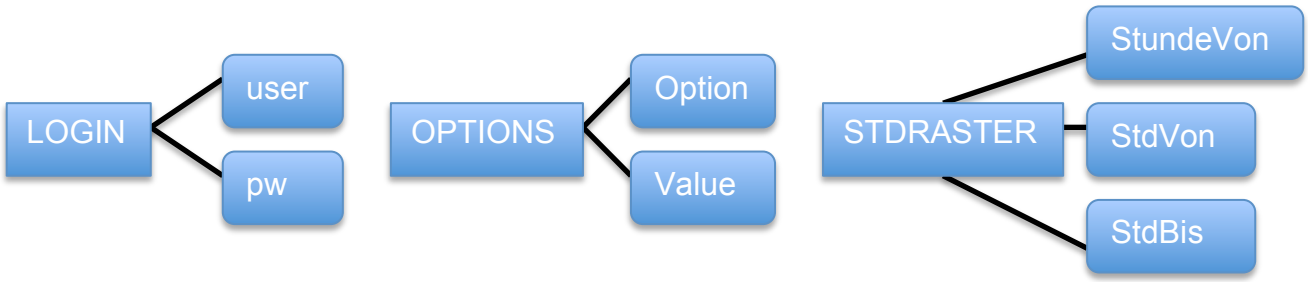

Abbildung 34: ER-Schemas für die Tabellen LOGIN OPTIONS und STDRASTER

Die Tabelle EKB ist das Kernstück der Datenbank. In dieser Tabelle sind zu jeder Stunde die Schülerdaten, Lehrerdaten, Fachdaten und Klassenbezeichnungen (Langsatz) gespeichert.

Der Primärschlüssel der Tabelle wurde aus Mandant (Schulnummer) und ID (laufende Nummer aus der Tabelle Klassenbuch des Schulverwaltungsprogramms) gebildet.

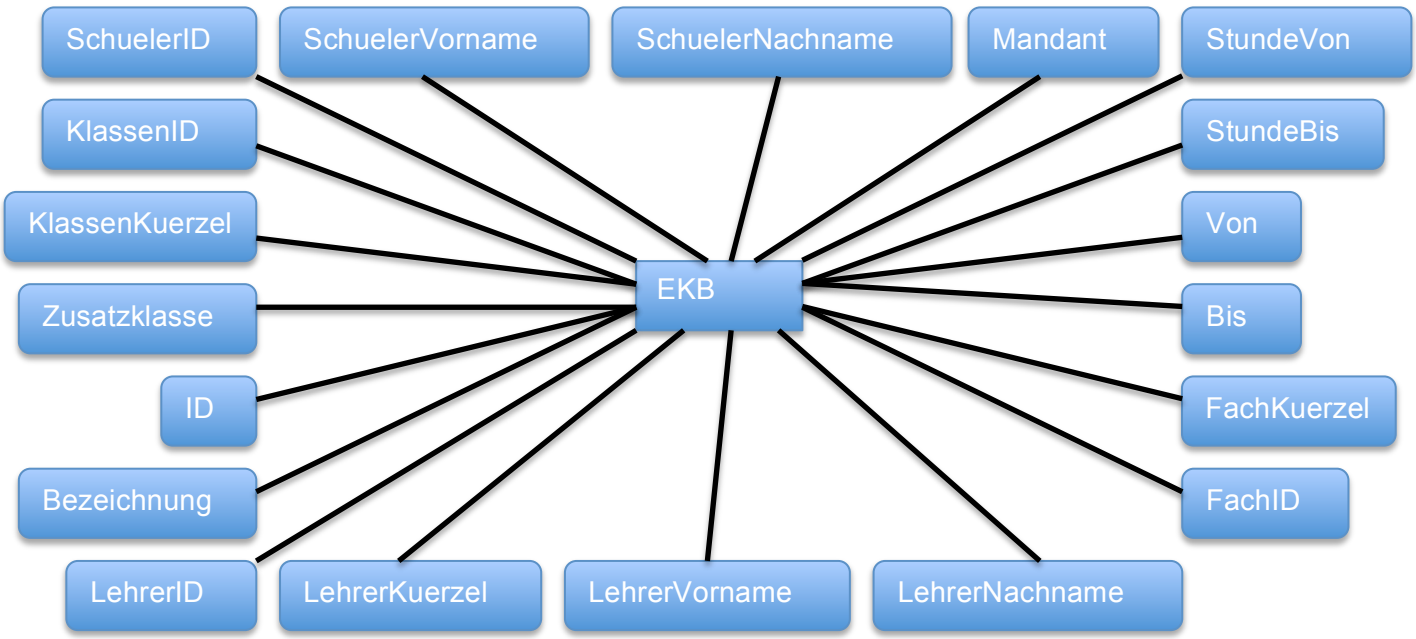

Abbildung 35: ER-Schema der EKB Tabelle

Im Feld Bezeichnung wird die Fachbezeichnung des Unterrichtsfaches abgespeichert. Die übrigen Felder der Tabelle sind selbsterklärend.

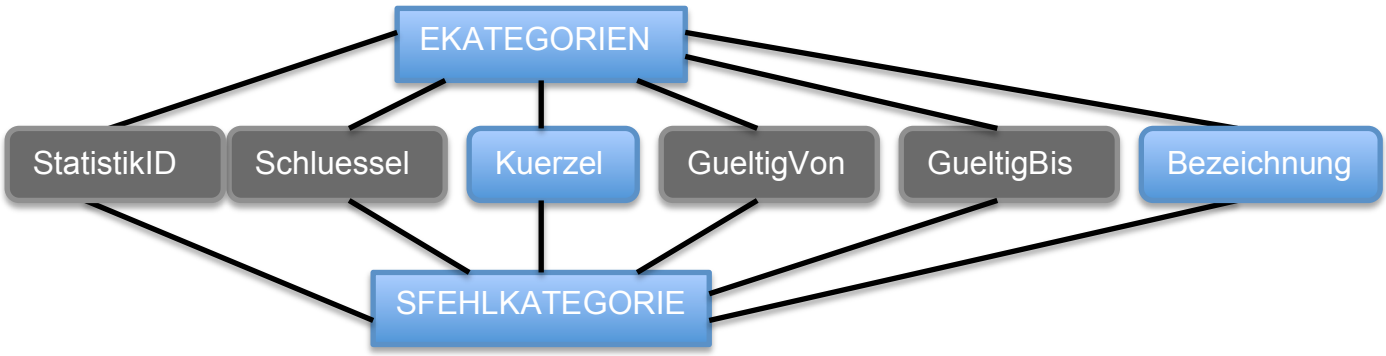

Abbildung 36: ER-Schema der Tabellen EKATEGORIEN und SFEHLKATEGORIEN

Die beiden Tabellen EKATEGORIEN und SFEHLKATEGORIEN entsprechen den Tabellen aus dem Schulverwaltungsprogramm. EKATEGORIEN enthält die Entschuldigungskategorien und die Tabelle SFEHLKATEGORIEN die

Schülerfehlzeitkategorien. Die Felder *StatistikID, Schluessel, GueltigVon* und *GueltigBis* werden zurzeit im Schulverwaltungsprogramm nicht.

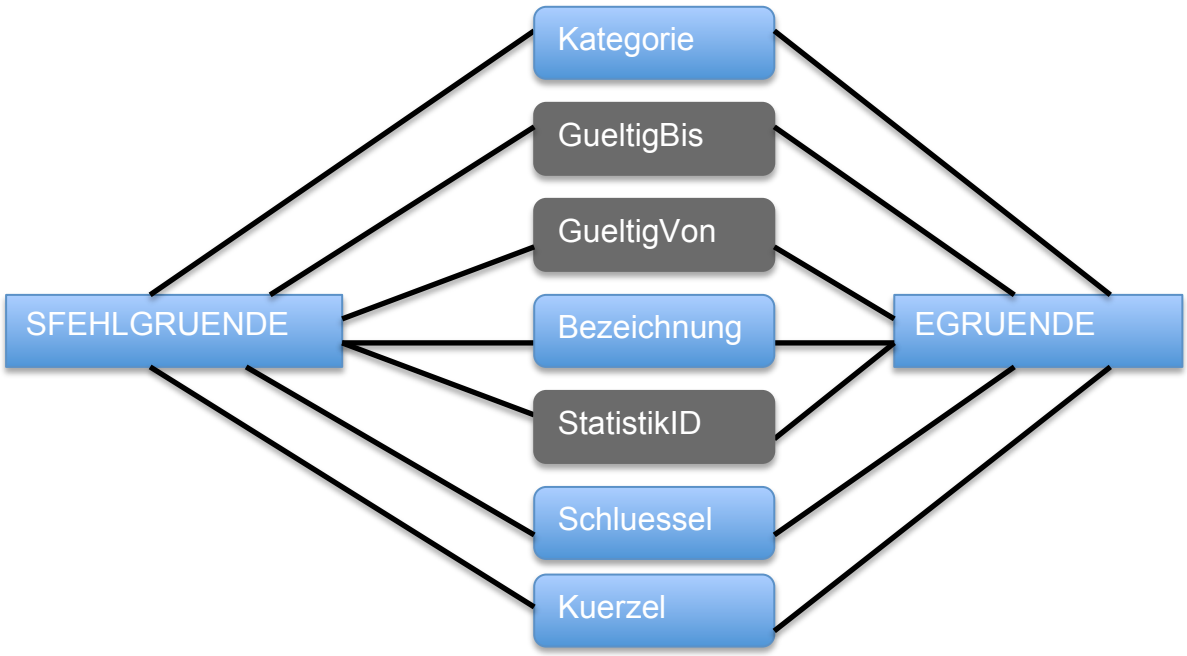

Abbildung 37: ER-Schema der Tabellen SFEHLGRUENDE und EGRUENDE

Die Tabelle SFEHLGRUENDE enthält die Schülerfehlgründe und die Tabelle EGRUENDE die Entschuldigungsgründe. Diese entsprechen wiederum den Schlüsseltabellen des Schulverwaltungsprogramms.

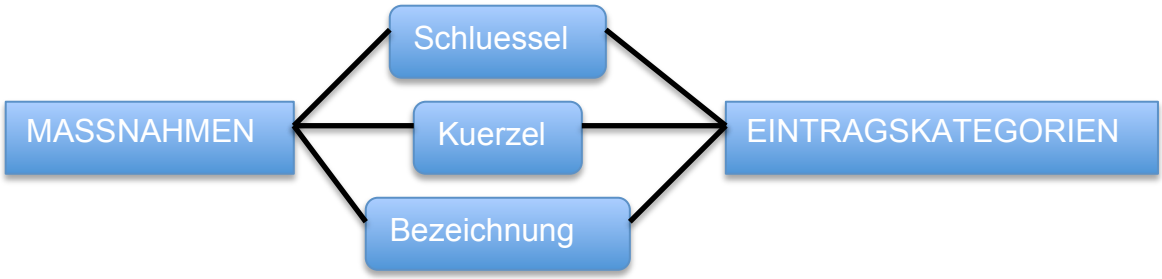

Abbildung 38: ER-Schemas der Tabellen MASSNAHMEN und EINTRAGSKATEGORIEN

Wie bei den beiden vorherigen Tabellen Abbildung 36: ER-Schema der Tabellen EKATEGORIEN und SFEHL handelt es sich bei den Tabellen MASSNAHMEN und EINTRAGSKATEGORIEN um Schlüsseltabellen des Schulverwaltungsprogramms. Die EINTRAGUNGSKATEGORIEN enthalten eine Liste zum Lern- und Sozialverhalten, während in MASSNAHMEN die möglichen entsprechenden pädagogischen Antworten vorgegeben sind.

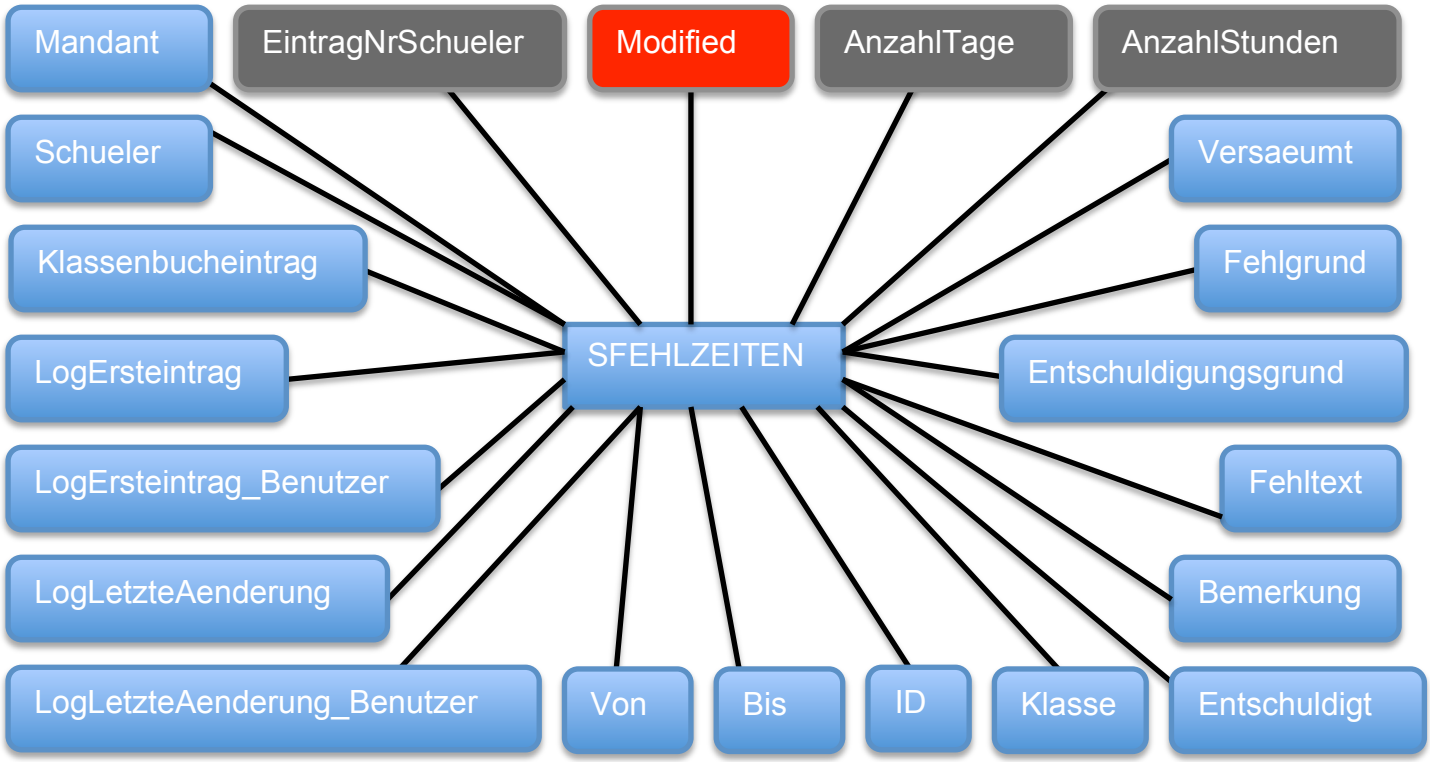

Abbildung 39: ER-Schema der Tabelle SFEHLZEITEN

In der Tabelle SFEHLZEITEN werden Daten von jeweils einer Schülerfehlzeit gespeichert. Die Struktur wurde wiederum entsprechend der Tabelle SchuelerFehlzeiten des Schulverwaltungsprogramms erstellt. Auffällig ist, dass beim Datenbankdesign der Firma Stüber eine Reihe von Feldern aufgenommen wurden, die offensichtlich erst für spätere Auswertungen und Berichte vorgesehen sind.

Für die Eintragungen stehen die Schlüsseltabellen EGRUENDE und SFEHLGRUENDE zur Verfügung. *Entschuldigungsgrund* und *Bemerkungen* sind

Freitext-Felder, die zur ergänzenden Beschreibung gedacht sind. Besonders erwähnenswert für die Implementierung ist das Feld ID. Dies ist eine vom Schulverwaltungssystem vergebene, laufende Nummer, die bei einer Neueingabe einer Fehlzeit auf dem mobilen Gerät zunächst den Wert NULL erhält. Nur Datensätze, bei denen das Feld ID den Wert NULL besitzt, werden bei der Synchronisation an die zentrale Datenbank zurückgegeben. Wird ein älterer, bereits synchronisierter Datensatz nachträglich verändert, wird ID wieder auf NULL gesetzt und dadurch nur dieser Datensatz zurückgeliefert. Gleiches gilt für die Tabelle SVERMERKE.

In der Tabelle SVERMERKE werden Informationen zum Lern- und Sozialverhalten eines Schülers gespeichert. Auch hier sind offensichtlich für einen späteren Report zusätzliche Felder aufgenommen, die in der aktuellen Version nicht benutzt werden.

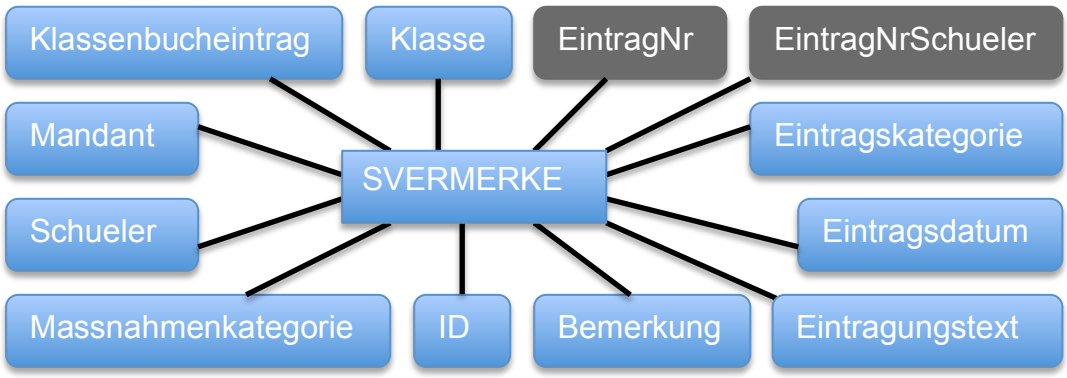

Abbildung 40: ER-Schema der Tabelle SVERMERKE

In Eintragungskategorie und Massnahmenkategorie werden die Kürzel der in Abbildung 38: ER-Schemas der Tabellen MASSNAHMEN und EINTRAGSKATEGORIEN dargestellten Tabellen eingefügt. Analog zu den Schülerfehlzeiten können auch hier, durch die Freitext-Felder *Eintragungstext* und *Bemerkung*, Ergänzungen aufgenommen werden. Die Bedeutung des Feldes ID entspricht dem gleichnamigen Feld der Tabelle SFEHLZEITEN.

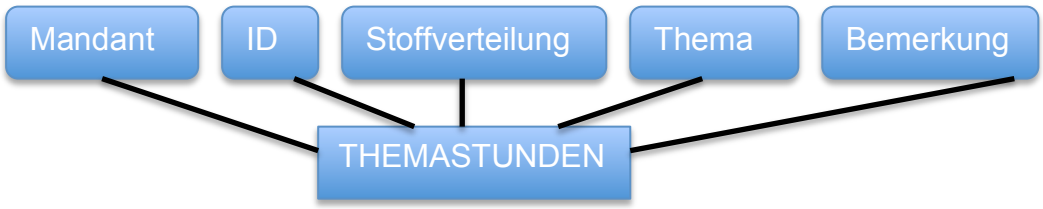

Abbildung 41: ER-Schema der Tabelle THEMASTUNDEN

Die Tabelle THEMASTUNDEN enthält zum jetzigen Zeitpunkt lediglich Informationen zum Stundenverlauf. Das Feld Stoffverteilung stellt eine Referenz zur Schlüsseltabelle STOFFVERTEILUNG dar, die zum jetzigen Zeitpunkt im Schulverwaltungsprogramm nicht gefüllt ist.

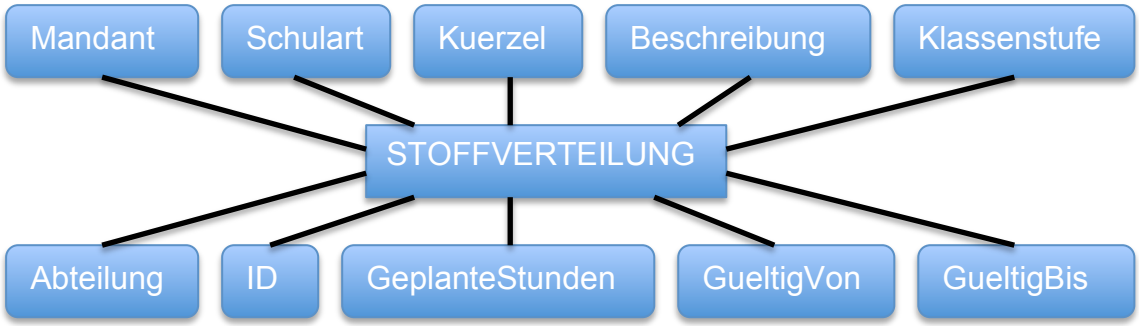

Abbildung 42: Die Tabelle Stoffverteilgung

Für die Implementierung des Prototyps wurden eigene Testdaten zu gängigen Themen aus der Mathematik, Naturwissenschaft und Sozialkunde hinterlegt.

In der Tabelle HAUSAUFGABEN sind Informationen über gestellte Aufgaben gespeichert. Diese ergänzen die Angaben zum Thema der Stunde durch die Anzeige der jeweils in der vorherigen Stunde gestellten Aufgaben.

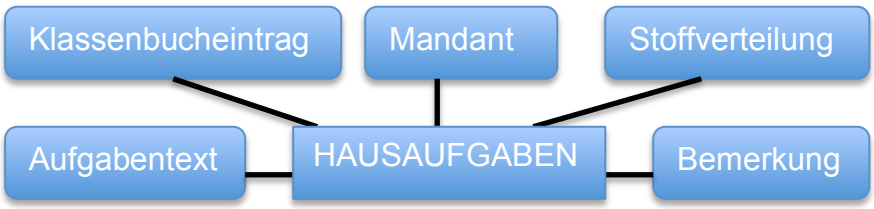

Abbildung 43: ER-Schema der Tabelle HAUSAUFGABEN

In der Tabelle PICS werden die Bilder der Schüler gespeichert. Neben der SchülerID wird auch der Mandant (Schulnummer) dort abgelegt.

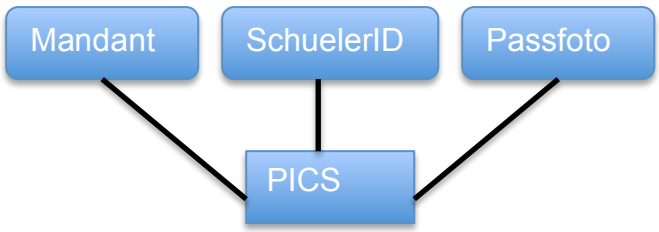

Abbildung 44: ER-Schema der Tabelle PICS

In Abbildung 45: ER-Übersicht der Relationen in Min-Max-Notation sind die Relationen entsprechend der Min-Max-Notation dargestellt (vgl. Kemper & Eickler, 2006, S. 44).

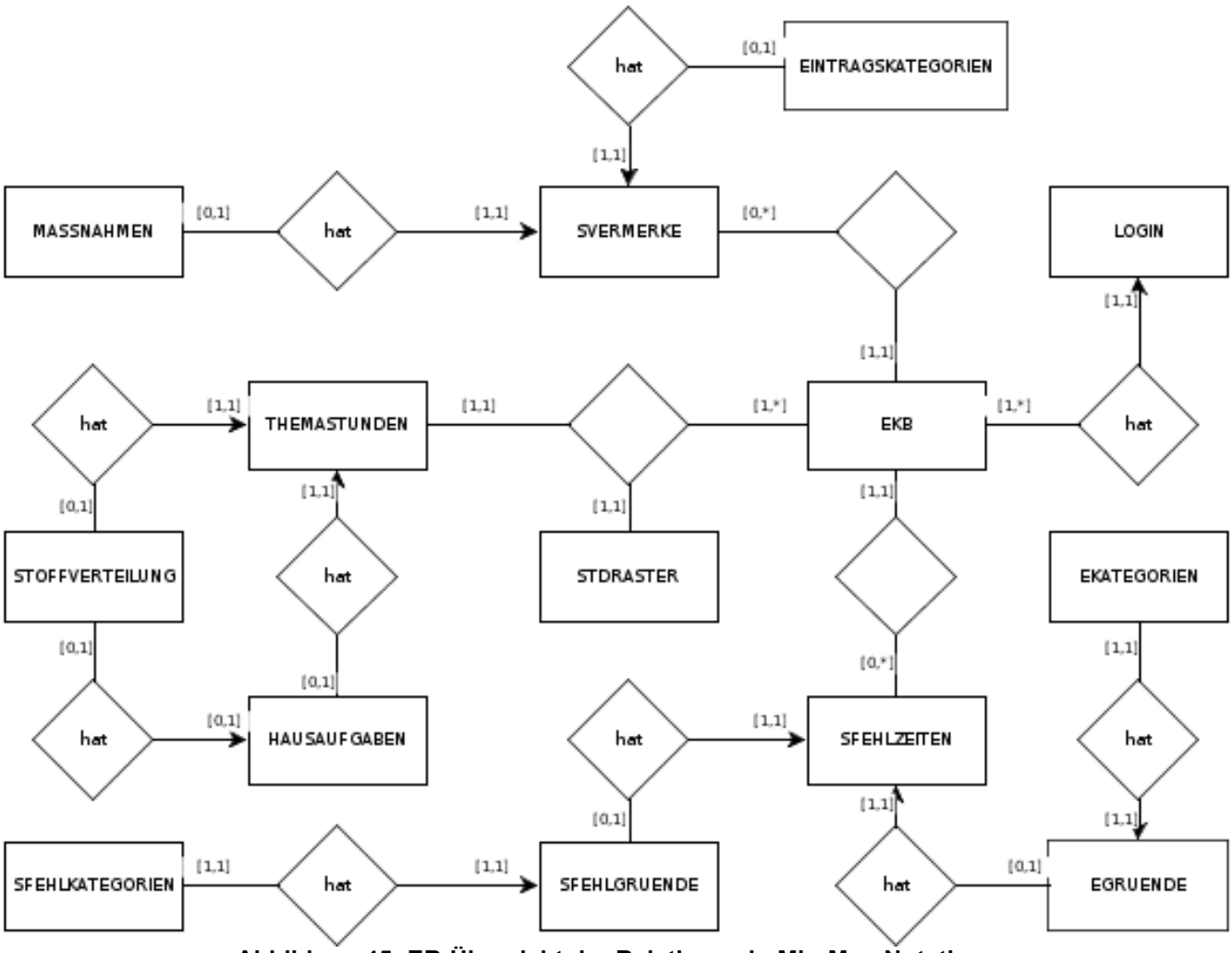

Abbildung 45: ER-Übersicht der Relationen in Min-Max-Notation

4.2.2 Asynchrone Datenübertragung

Bei einem Konzept, in dem die Informationen direkt nach der Erfassung auf dem Server gespeichert werden, wird eine dauerhafte Netzwerkverbindung benötigt. Dies ließe sich zum Beispiel durch den Einsatz von *Universal Mobile Telecommunications System* (UMTS) Geräten oder der flächendeckenden Einführung eines Schul-WLANs realisieren.

Beide Methoden haben jedoch gravierende Nachteile: Während im Falle einer UMTS-Anbindung der mobilen Geräte Kosten anfallen würden, für deren Übernahme durch die SfBWuG weder Regularien noch finanzielle Spielräume vorhanden sind, ist

bei einer Datenbearbeitung ausschließlich im Schulnetz die gewünschte Nachbereitung von Informationen außerhalb dieses Netzes nicht möglich (vgl. u.a. (Hansen, 2011) ; (ifib, 2011)).

Für den Prototypen wurde daher von der SfBWuG die Möglichkeit geschaffen, sowohl aktuelle Daten aus dem Schulverwaltungsprogramm auf das mobile Gerät zu laden, Daten aus der Unterrichtssituation zu erfassen und zurück zu schreiben, als auch die Bearbeitung von bereits erfassten und gespeicherten Daten nachträglich außerhalb des Schulnetzes zu ermöglichen.

Webservices sind prinzipiell geeignete Methoden, um Daten zwischen Endgeräten und Servern auszutauschen. Im Rahmen dieser Arbeit musste jedoch auf die Entwicklung eines Webservices verzichtet werden, da aufgrund des relationalen Datenmodells von Magellan, der Schulverwaltungssoftware, allein für das Abrufen der Daten zu einer Klasse eine View über 7 Tabellen für einen Datensatz eines Schülers hätte ausgeführt werden müssen. Bei ersten Tests erwies sich dies trotz Optimierungsversuchen durch die SfBWuG aufgrund der Laufzeit bei einem Datenaustausch von nur drei Unterrichtswochen als völlig ungeeignet. Um einen performanteren Webservice implementieren zu können, hätte das Datenbankmodell der Firma Stüber grundlegend verändert werden müssen.

Daher wurde von der SfBWuG eine Schnittstelle zur Firebird-Datenbank bereitgestellt. Dabei findet der Datenaustauch in JavaScript Object Notation (JSON), nach einer vordefinierten Struktur statt, die im Anhang dokumentiert ist. Um eine effiziente Verarbeitung auf den mobilen Geräten zu erreichen, wurde mit Ausnahme der Schlüsseltabellen von einer 1:1 Übertragung der Tabellen des Schulverwaltungsprogramms abgesehen. Für die Tabelle EKB wurde durch Re-Normalisierung eine Struktur erstellt, in der zu jeder Unterrichtsstunde die Schülerdaten, die Lehrerdaten, Fachdaten und Klassenbezeichnungen zusammengefasst wurden (Langsatz). Dieser Weg wurde gewählt, um auf den mobilen Geräten mit einem einzigen SQL-Datenbankzugriff den kompletten Datensatz einer Unterrichtsstunde mit allen relevanten Daten für die Anzeige bereitstellen zu können.

4.3 Beschreibung der Benutzungsoberfläche

Für die Erstellung der Benutzungsoberfläche wurde versucht, eine Unterrichtssituation hinsichtlich eines fiktiven Arbeitsablaufes abzubilden. Aus diesem Grunde werden inhaltlich zusammenhängende Informationen auf ein und derselben Oberfläche angezeigt. Damit ergibt sich der in Abbildung 46: Programmfluss des elektronischen Klassenbuchs beschriebene Programmfluss.

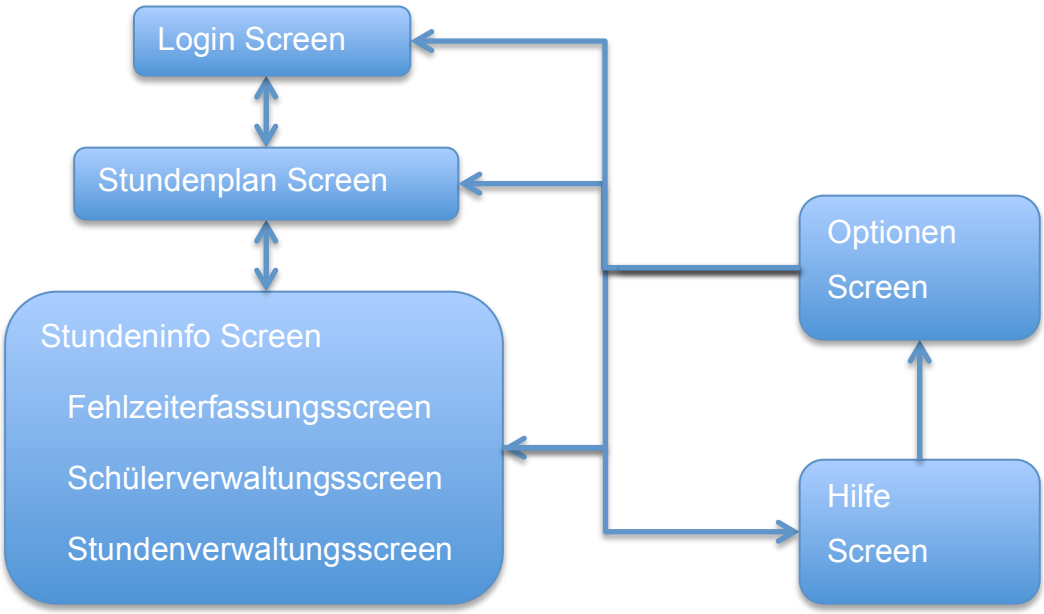

Abbildung 46: Programmfluss des elektronischen Klassenbuchs

Alle statischen, grafischen Elemente sind in der Datei *index.html* definiert. Die Gestaltung von grafischen Elementen, deren Design nicht durch Elemente aus Abbildung 10: Realisierbarkeit der nativen iOS Elemente mit dem DojoToolkit erfasst werden, sind in der CSS-Datei *custom.css* definiert.

4.3.1.1 Login-Screen

Die Applikation kann nach dem Starten nur fortgesetzt werden, wenn der Benutzer sich mit gültigen Anmeldedaten authentifiziert hat.

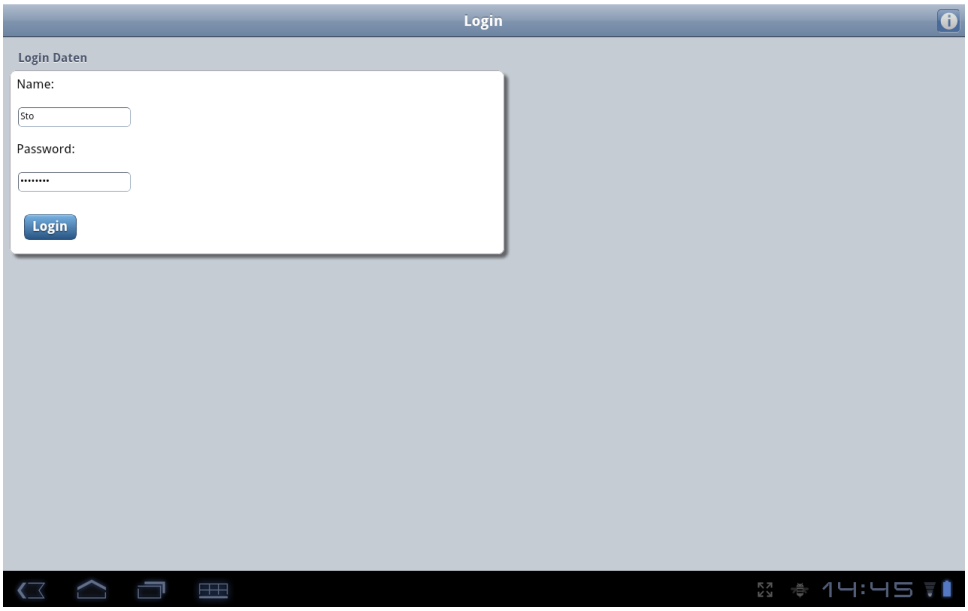

Abbildung 47: Login Screen auf einem Android System

Bei der Erstellung des Login Screens werden sechs verschiedene Objekte verwendet. Dabei kommen folgende in Kapitel 2.3.2 bereits beschriebene DojoToolkit-Elemente für die Erstellung von Benutzungsoberflächen zur Anwendung:

1. Element I, Navigationsleiste
2. Element XV, Info-Button
3. Element XXVI, Textfeld
4. Element XXVI Textfeld
5. Element VI Table View
6. Element XX Rounded Rectangle Button

4.3.1.2 Stundenplan-Screen

Um durch die Unterrichtsstunden eines Lehrers navigieren zu können, liegt es nahe, dieses in Form eines Stundenplans zu realisieren. Da es kein Element für eine solche Darstellung in den Style Guides gibt, wurde sich an die gängige Darstellung von Stundenplänen gehalten.

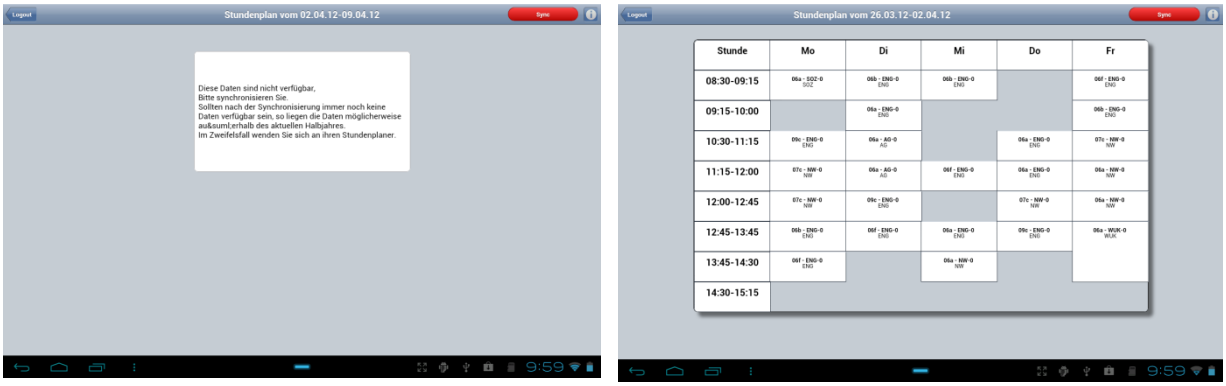

Abbildung 48: Stundenplan Screen auf einem Android System

Trotzdem werden wie in Abbildung 48: Stundenplan Screen auf einem Android System dargestellt einige der Elemente aus Kapitel 2.3.2 verwendet:

1. Element I, Navigationsleiste
2. Element XV, Info-Button
3. Element I, Navigationsleiste (Button)

Ein Nebeneffekt dieser Darstellung ist nicht nur die gleichzeitige Bereitstellung des Stundenplans für den Nutzer, sondern, bei tagesaktueller Synchronisation, auch die automatische Anzeige von Vertretungsstunden in der Übersicht.

4.3.1.3 Stundenverwaltungs-Screen

Durch die Nutzung einer Tab-Bar bietet sich die Möglichkeit, thematisch zusammenhängende Eingaben zugänglich zu machen. Für die Stundenverwaltung wird eine Fehlzeiteneingabe angezeigt, von der eine Ergänzung/Korrektur, sowie eine Eingabe zum Lern- und Sozialverhalten ermöglicht wird. Als drittes Element ist über die Tab-Bar eine Beschreibung zur aktuellen Stunde möglich.

Der Stundenverwaltungs-Screen besteht aus den folgenden Elementen aus Kapitel 2.3.2:

1. Element I, Navigationsleiste
2. Element XV, Info-Button
3. Element I, Navigationsleiste (Button)
4. Element III, Tab Bar

4.3.1.3.1 Fehlzeitenerfassung

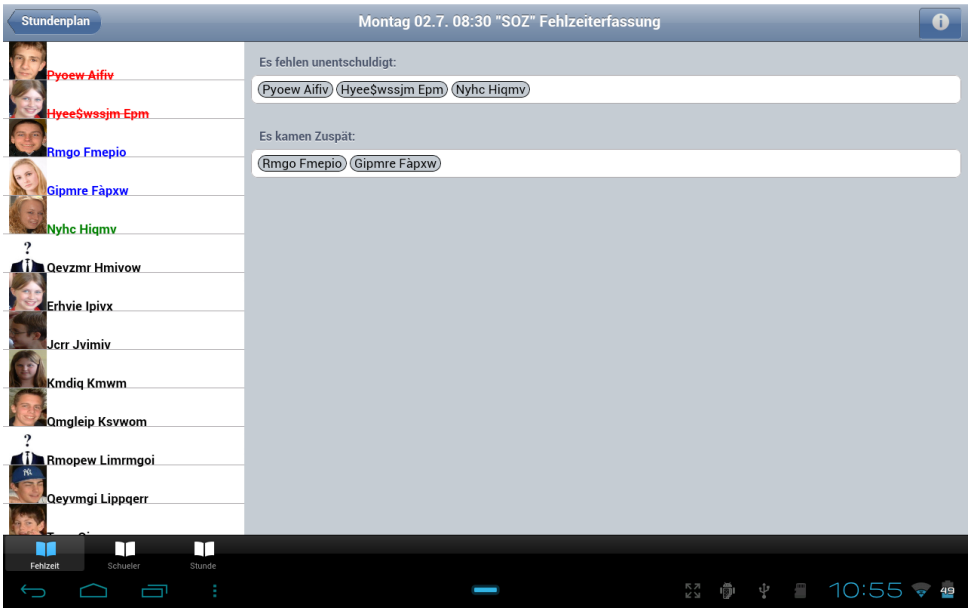

Abbildung 49: Fehlzeitenerfassung

Auf der Benutzungsoberfläche der Fehlzeitenerfassung werden Standardeintragungen in Bezug auf die Fehlzeiten von Schülern vorgenommen. Dabei wird ein Schüler als *unentschuldigt abwesend* oder als *verspätet erschienen*

eingetragen. Dies wird, wie in Abbildung 49: Fehlzeitenerfassung ersichtlich, durch folgende Elemente realisiert:

1. Element V Split View
2. Element VI Table View

Die Aufzählung der Namen im rechten Teil des Split-Views wurde an der iOS Umsetzung von Namenslisten - zum Beispiel beim Schreiben von SMS - angelehnt, da sich in den Guidelines hierfür kein explizites Element oder eine adäquate Beschreibung befindet.

4.3.1.3.2 Schülerübersicht

In diesem Teil der Benutzungsoberfläche werden alle Eintragungen eines Schülers für die aktuelle Unterrichtsstunde aufgelistet und können bearbeitet werden.

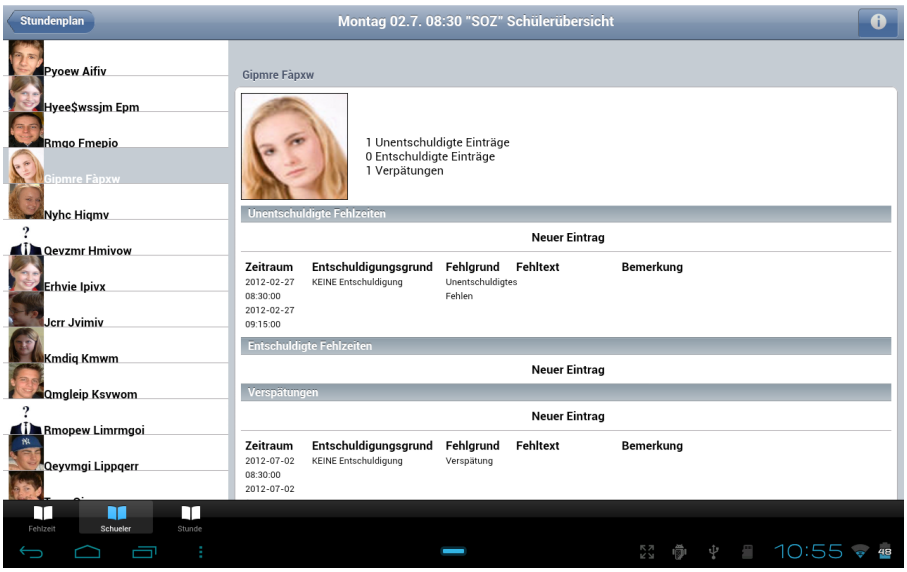

Abbildung 50: Schülerübersicht

Zur Darstellung werden die Design-Elemente V *Split View* und VI *Table View* angewandt. Insbesondere können von dieser Stelle aus Entschuldigungen oder Ergänzungen zu den Verspätungs- oder Fehlgründen nachgetragen werden. Hierzu werden die in Abbildung 50: Schülerübersicht dargestellten ModalViews genutzt, die auch für die Eingabe und gegebenenfalls zur Modifizierung von Einträgen zum Lern- und Sozialverhalten des Schülers dienen.

4.3.1.3.3 Thema der Stunde

Innerhalb dieser Oberfläche sind alle Informationen zusammengefasst, die die aktuelle Stunde kennzeichnen und perspektivisch für die Entwicklung und Sicherung der Qualität von Schule und Unterricht relevant sein können. Dazu gehört eine Beschreibung der Stunde, sowie der Hausaufgaben, sowohl zur aktuellen Stunde als auch zur nächsten Stunde. Als weitere Eingabe kann hier die Einordnung in den Stoffverteilungsplan des schulspezifischen Curriculums vorgenommen werden. Zur Darstellung wird das Element VI *Table View* angewandt.

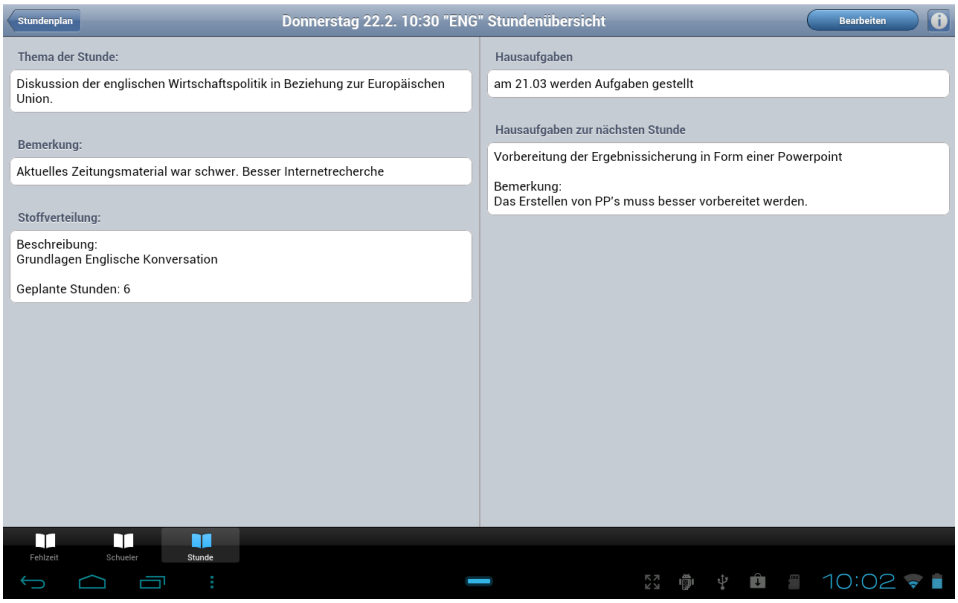

Abbildung 51: Stundenübersicht

4.3.1.4 ModalViews

Um Eingaben oder Korrekturen in der Schülerübersicht und Themenübersicht vornehmen zu können, wird das Element XI *Modal View* mit den jeweiligen kontextabhängigen Eintragungsmöglichkeiten verwendet. Für das Eintragen von Werten der Schlüsseltabellen wird ein zweiter *Modal View* angezeigt, in dem ein Picker (Element XVIII) eingebettet ist.

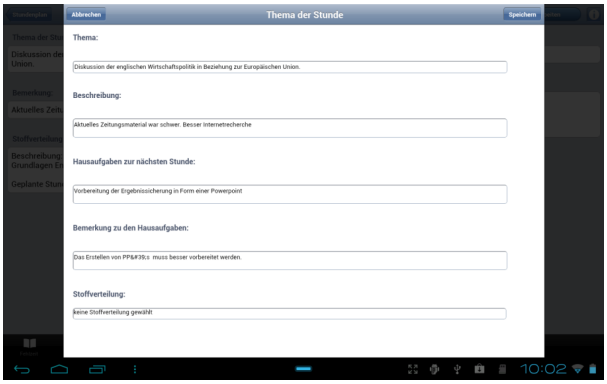

Abbildung 52: Modal View zum Lern und Sozialverhalten

Abbildung 53: Modal View zur Stundenübersicht

4.3.1.5 Hilfe und Optionen

Sowohl die Hilfe als auch die Optionen werden in einem Popover Fenster angezeigt. Dieses ist entsprechend dem Element IV *PopoverView* entwickelt. Außerdem kommt noch Element VI *Table View* für die Strukturierung des Inhalts zur Anwendung.

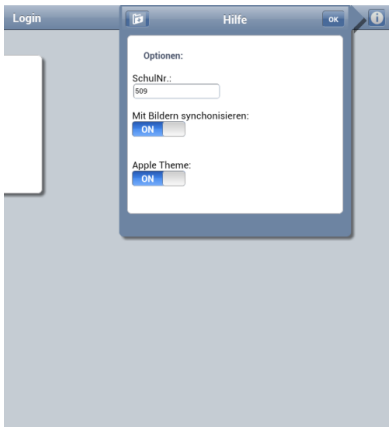

Abbildung 54: PopoverView: Optionen

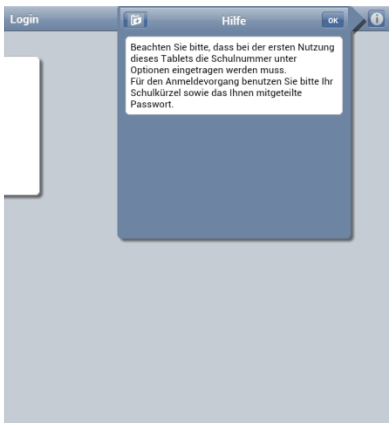

Abbildung 55: PopoverView: Hilfe

4.4 Die Implementierung

Die nun ausführlich diskutierten Rahmenbedingungen und Design-Vorgaben für eine prototypische Entwicklung eines digitalen Klassenbuchs werden im nächsten Schritt umgesetzt. Die Hilfsmittel werden aufgeführt und ausgewählte Programmabschnitte erläutert.

4.4.1 Die Entwicklungsumgebungen

Durch die Entscheidung für PhoneGap als Framework ist ein leistungsfähiger Editor mit Unterstützung für die Erstellung von HTML, CSS und JavaScript-Dateien zweckmäßig. Für die Erstellung der Applikationen für das iPad ist neben PhoneGap ein Macintosh mit OSX Lion als Betriebssystem zwingende Voraussetzung. XCode ist die einzige Entwicklungsumgebung, die Apple für die Erstellung von Programmen offiziell unterstützt und muss auch für die Erstellung von Anwendungen mit PhoneGap genutzt werden. Um PhoneGap für Android Systeme nutzen zu können, wird von Google Eclipse als Entwicklungsumgebung empfohlen. Da Eclipse auch für OSX verfügbar ist, wurde dieses zur Entwicklung der Anwendung genutzt.

Bei der Entwicklung stellte sich heraus, dass die Debug-Informationen innerhalb der Eclipse-Umgebung wesentlich ausführlicher sind, als in XCode. Deshalb wurde XCode nur zum Kompilieren der finalen Applikation für das iPad verwendet, während als Editor für HTML, CSS und JavaScript Eclipse auf dem Macintosh genutzt wurde. Für die Erstellung der Applikation für Android-Systeme wurde das Android-SDK und der Android-Device-Manager in Eclipse eingebunden.

Anzumerken ist in diesem Zusammenhang, dass eine Entwicklung für Android-Geräte auf einem Windowsbetriebssystem durch das Fehlen der jeweiligen gerätespezifischen USB-Treiber erheblich erschwert wird, da diese in der Regel nur schwer von den jeweiligen Herstellerseiten der Android-Geräte erhältlich sind.

4.4.2 Das DojoToolkit

Auf Grund der in Kapitel 2.3.2 diskutierten Kriterien ist die Bibliothek DojoToolkit die beste Wahl. Dennoch wurde vorab geprüft, ob diese auf dem iPad und Android-Geräten fehlerfrei funktioniert. Im Unterschied zum iPad stellte sich überraschend heraus, dass für Android-Geräte das dynamische Nachladen von DojoToolkit-

Modulen nicht unterstützt wird (vgl. Keese, 2011). Die einzige Alternative zu dieser Methode, besteht darin, ein *custombuild* mit nur den Elementen für die Applikation aus dem DojoToolkits zu erstellen. Diese Vorgehensweise hat den Vorteil, dass die Applikation lediglich den benötigten Funktionsumfang besitzt und damit die Performance verbessert wird (vgl. Russell, 2008, S. 396).

Zur Erstellung eines passenden *custombuild* müssen alle relevanten Dojo-Komponenten in ein einziges File integriert werden. Dafür stellt das DojoToolkit ein Buildsystem zur Verfügung (vgl. Russell, 2008, S. 396). Um unter Linux beziehungsweise unter OSX das Skript *build.sh* nutzen zu können, muss eine sogenannte Profildatei erstellt werden. In der src-Version des DojoToolkits befindet sich unter *util/buildscripts/profiles/* die Profildatei *baseplus.profile.js*, die als Grundlage für das *custombuild* dieser Arbeit dient. Die so entstandene Profildatei *myPhoneGap.profile.js* ist im Folgenden aufgeführt.

Beachtet werden muss, dass ab der Version 1.7 von dojoToolkit als selectorEngine *acme* verwendet wird. Gegenüber der anfangs benutzten 1.6 Version von DojoToolkit wurde das komplette Dateiverwaltungssystem ab dieser Version verändert, sodass dieses neue *custombuild* erstellt werden musste.

Hier folgt der Code aus myPhoneGap.profile.js

```
dependencies = {
selectorEngine: "acme",
layers: [
        {
        // This is a specially named layer, literally 'dojo.js'
        // adding dependencies to this layer will include the modules
        // in addition to the standard dojo.js base APIs.
        name: "dojo.js",
        dependencies: [
                "dijit._Widget",
                "dijit._Templated",
                "dojo.fx",
                "dojo.NodeList-fx",
                //this wasn't included in the standard build but necessary
                "dojo._firebug.firebug",
                //my used dojo requirements
                "dojox.mobile._base",
                "dojox.mobile._ScrollableMixin",
                "dojox.mobile.parser",
                "dojox.mobile",
                "dojox.mobile.Button",
                "dojox.mobile.SwapView",
```

```
                "dojox.mobile.ScrollableView",
                "dojox.mobile.TabBar",
                "dojox.mobile.SpinWheelTimePicker",
                "dojox.mobile.TextBox",
                "dojox.mobile.ExpandingTextArea",
                "dojox.mobile.compat"
                ]
        }
        ],

prefixes: [
        ["dijit", "../dijit" ],
        ["dojox", "../dojox" ]
        ]
}
```

Eine weitere Schwierigkeit des DojoToolkits in der derzeitigen Version 1.7 liegt darin, dass, wie in Kapitel 2.3.3 beschrieben, noch nicht alle Objekte problemlos genutzt werden können. So sind auch die *SpinWheels*, welche für das Element XVIII relevant sind, im Gegensatz zur Dokumentation nicht dynamisch veränderbar. Um dieses Problem zu umgehen wurde folgende Lösung gefunden:

Bevor Daten einem solchen *SpinWheel* zugewiesen werden, wird dieses zunächst gelöscht und anschließend mit gleicher *spinWheelID* neu erstellt. Um nun Daten in den neuen *SpinWheelSlot* einfügen zu können, muss auch dieses Element neu mit den Daten erstellt werden. Diese Elemente werden erst nach ihrem vollständigen Aufbau an einer Stelle innerhalb der View mit *spin.placeAt* eingefügt.

Erstellung eines SpinWheels:

```
dijit.byId('spinWheelID').destroy();
   var spin = new dojox.mobile.SpinWheel({'id':'spinwheelID',});
   var slot = new dojox.mobile.SpinWheelSlot({'labels':Daten});
   spin.addChild(slot);
   spin.placeAt('parentElement_des_SpinWheels');
   spin.startup();
```

4.4.3 Die Implementierung ausgewählter Funktionen

Generell wurden die Funktionen in Dateien nach Sachzusammenhang gegliedert. Alle netzwerkspezifischen Funktionen wurden in der Datei *network.js* zusammengefasst, während alle Datenbankoperationen über Funktionen in der Datei

DB.js ausgeführt werden. Die grafischen Objekte der Benutzungsoberfläche bekommen ihre Funktionalität durch Skripte, die in den jeweiligen Dateien mit dem Namen der zugehörigen Objekte abgelegt wurden. So finden sich die Funktionen zur Erstellung des Stundenplans in *plan.js*.

4.4.3.1 Die Synchronisation mit dem Schulserver

Asynchronous JavaScript and XML (AJAX) ist die Methode der Wahl, um Inhalte auf Webseiten nachzuladen. Zwar muss bei einem AJAX-Request generell eine Fallunterscheidung zur Feststellung des benutzten Browsers erfolgen, jedoch kann, solange Tablett-Systeme mit *Windows 8 Metro* nicht verfügbar sind, eine eigene Browserweiche entfallen.

Ein AJAX-Aufruf erfolgt nach dem hier angeführten Mechanismus:

```
var ajax= new XMLHttpRequest();
 ajax.open("GET", 'http://localhost/getText.php', true);
ajax.send();
ajax..onreadystatechange = function(content){
   if(ajax.readyState==4 && ajax.status==200){
      alert(ajax.responseText);
   }
}
```

Aus Gründen der zukünftigen Kompatibilität, auch mit *Windows 8 Apollo*, bietet es sich aber an, auf die im DojoToolkit bereitgestellte Bibliothek zurückzugreifen.

Ajax Funktionalität von (The Dojo Foundation, 2012):

```
var xhrArgs = {
   url: "{{dataUrl}}dojo/LICENSE",
   handleAs: "text",
   load: function(data){
     // Replace newlines with nice HTML tags.
     data = data.replace(/\n/g, "<br>");

     // Replace tabs with spaces.
     data = data.replace(/\t/g, "   ");

     targetNode.innerHTML = data;
   },
   error: function(error){
```

```
    targetNode.innerHTML = "An unexpected error occurred: " + error;
   }
  }

  // Call the asynchronous xhrGet
  var deferred = dojo.xhrGet(xhrArgs);
});
```

Durch die Nutzung von PhoneGap tritt noch eine weitere Besonderheit auf, die als Cross-domain-Problem bezeichnet wird (vgl. Simpson, 2011). Diese Sicherheitsrichtlinie greift, wenn Daten domänenübergreifend abgerufen werden sollen. Das Problem konnte analog zum Apache-Webserver durch Aufnahme der IP-Adresse beziehungsweise des vollständigen DNS-Bezeichnung des Webservers in die jeweilige plattformabhängige *Whitelist* von PhoneGap gelöst werden.

Ein Unterschied zur Nutzung von klassischen Webseiten besteht im Datenaustausch/Aufruf einer Webseite. Dieses kann nur dann fehlerfrei durchgeführt werden, wenn der entsprechende Server erreichbar ist. Im Gegensatz dazu steht auf dem Tablett-System die komplette Applikation einschließlich der synchronisierten Daten dauerhaft zur Verfügung, so dass entsprechend der Spezifikation des ifib, die Bearbeitung **ohne** direkten Zugriff auf das Schulnetz erfolgen kann (vgl. ifib, 2011). Deswegen wird der Datenzugriff einschließlich des Austausches mit dem Server durch eine besondere Fehlerbehandlung und die Speicherung auf dem Tablett-System ergänzt.

Zur Aufbereitung der Daten für den Datenaustausch werden die Funktionen aus der Datei *synchronize.js* verwendet. Hierdurch werden Zeichensatzkonvertierungen und die Rückumwandlung von Umlauten und ausgewählten Sonderzeichen vorgenommen und für die Übermittlung im JSON-Format vorbereitet. Der Aufbau der JSON-Zeichenkette erfolgt in der Funktion *sync.resync()*. Der Austausch mit dem Schulserver erfolgt mit Hilfe der Funktionsaufrufe *network.serverLogin()*, *network.sync()* und *network.getPics()* mit Hilfe von Parametern innerhalb dieser Funktionen. Bevor Daten bearbeitet werden können, wird der Zugriff mit Hilfe von *network.serverLogin()* überprüft. Diese Funktion wird in Kapitel 4.4.3.3 das Einloggen gesondert beschrieben.

Sollen Daten mit dem Server aktualisiert werden, wird serverseitig durch *network.sync()* im ersten Schritt das PHP-Skript *resync.php* angesprochen. Dabei werden als erstes an den Server neu erstellte beziehungsweise geänderte Datensätze übermittelt und in der Datenbank auf dem Schulserver gespeichert. Im zweiten und dritten Schritt werden die Skripte *syncStart.php* und *syncEnd.php* mit Hilfe der Parameterliste *mandant, username, password, von, bis* aufgerufen. Damit werden alle benötigten Datensätze innerhalb des Zeitintervalls zwischen *von* und *bis* angefordert. Diese Daten werden über ein verschlüsseltes WLAN im JSON-Format übermittelt, dessen Struktur im Anhang dargestellt ist.

Für die optionale Übermittlung von Bildern mit *network.getPics()* ist jeweils ein Aufruf je Schüler erforderlich. Diese Lösung war notwendig, da bei ersten Tests bereits bei einer Klassenstärke von nur 24 Schülern mit jeweils einem Bild der Transfer im JSON-Format mit einem Speicherfehler terminierte. Zur Aufbereitung der Daten für die Speicherung in der lokalen Datenbank werden die Funktionen aus der Datei *synchronize.js* verwendet. Hierdurch werden Zeichensatzkonvertierungen und die Rückumwandlung von Umlauten und ausgewählten Sonderzeichen vorgenommen.

4.4.3.2 Die Speicherung in der lokalen Datenbank

Um die aufbereiteten Daten in der lokalen Datenbank des Tablett-Systems speichern zu können, wird bereits beim Start der Applikation die Funktion *DB.init()* aufgerufen.

Hierdurch wird zunächst die Datenbank geöffnet und mit der Funktion *DB.createTables()* alle benötigten Tabellen erstellt, falls diese noch nicht vorhanden sind. Diese Überprüfung ist auch nach der Erstinstallation notwendig, da im Unterschied zu iOS-Geräten auf Android-Tabletts alle Daten einer Applikation jederzeit von dem Benutzer gelöscht werden können. Diese unterschiedliche Funktionalität wird im Abschnitt 5.1 aufgegriffen.

Initialisierung der Datenbank in DB.js:

```
DB.initDB = function(){
        console.error("try Init DB");
        DB.db = window.openDatabase("ekb", 1.0, "ekb", DB.DB_SIZE);
        console.error("DB initiated");
        DB.db.transaction(DB.createTables, DB.error);
}
```

Um einen Datensatz zu speichern, wird durch die PhoneGap-Funktion *executeSql(statement, errorfunction)* das jeweilige SQL-Statement ausgeführt. PhoneGap nutzt Transaktionen, sodass das eigentliche execute-Statement an die Funktion *transaction(function, errorfunction, succsessfunction)* mit dem Parameter *function* übergeben werden muss. Dabei ist *succsessfunction* optional, während *errorfunction* als Funktion definiert sein muss. Alle Datenbank Zugriffe sind in der Datei *DB.js* mit dem Objekt DB zusammengefasst. Dadurch erfolgt eine Kapselung der PhoneGap API vom Rest des Quellcodes.

4.4.3.3 Das Einloggen

Um Zugriff auf die Daten der Applikation zu erhalten, muss eine Authentifizierung des Benutzers erfolgen. Nach Eingabe des Benutzernamens und des Passwortes wird durch Tippen auf die Schaltfläche Login zunächst überprüft, ob die Nutzerdaten innerhalb der lokalen Datenbank enthalten sind (login.lokalLogin). Ist dies der Fall, wird eine Freigabe für den Zugriff erteilt und es kann lokal gearbeitet werden. Ist dies nicht der Fall, werden die Daten mit dem Server abgeglichen (network.login). Sind diese Daten korrekt, werden sie in die lokale Datenbank übernommen und der Zugriff wird freigegeben. Dies wird durch die Funktionen *login.lokalLogin()*, *network.login()*, *DB.load('login')* und *DB.save('login')* realisiert.

4.4.3.4 Der dynamische Stundenplanaufbau

Die für die Darstellung des Stundenplans notwendigen Funktionen werden in der Datei *plan.js* zusammengefasst. Um einen Stundenplan mit der korrekten Anzahl an Stunden dynamisch anzeigen zu können, wird aus der Datenbank die maximale Anzahl der Stunden pro Tag an der jeweiligen Schule aus der Abbildung 34: ER-Schemas für die Tabellen LOGIN OPTIONS und STDRASTER geladen. Daraus wird berechnet wie groß die einzelnen Terminfelder sein dürfen, um eine optimale

Darstellung auf den Bildschirm zu erreichen. Die Daten für den eigentlichen Stundenplan werden aus der Tabelle EKB extrahiert. Hier finden sich in den Feldern *Von, StundeVon, StundeBis, FachKuerzel*, und *KlassenKuerzel* alle Angaben, um den Stundenplan aufbauen zu können. Dies wird durch die Funktion *plan.setPlan()* realisiert.

Da es sich bei einem wochenbasierten Stundenplan um eine im Prinzip endlose Abfolge von Wochen handelt, musste ein besonderes Konzept zur Darstellung entwickelt werden. Unabhängig davon, ob Stundenplandaten vorhanden sind, werden drei Views für ein Drei-Wochen-Intervall realisiert. Die sichtbare View ist dabei immer die mittlere View, während rechts und links jeweils eine nicht sichtbare View liegt. Dies ermöglicht nicht nur das Scrollen des Bildschirmes durch ein Swipe-Event, sondern gewährleistet auch ein flüssiges blättern in beide Richtungen.

Wird zum Beispiel nach rechts geblättert, so wird auf die vormals sichtbare View zur rechten View. Diese, bisherige rechte View wird gelöscht. Damit die nun sichtbare View wieder zur mittleren View werden kann, wird auf der linker Seite eine View aufgebaut und gegebenenfalls mit Daten gefüllt. Sind keine Daten vorhanden, wird ein erklärender Text eingeblendet. Bei einem Swipe-Event nach links, läuft der Vorgang entsprechend. Dies ist in der Funktion *plan.handlePlan()* realisiert.

Durch einen Touch-Event auf eine Unterrichtsstunde in der Stundenplanübersicht, wird die Fehlzeitenerfassung (Abbildung 49: Fehlzeitenerfassung) des Stundenverwaltungs-Screens aufgerufen.

4.4.3.5 Die Fehlzeitenerfassung

Um Fehlzeiten oder Verspätungen möglichst einfach und schnell erfassen zu können, wird durch ein einfaches Touch-Event auf ein Element in der Liste der Schüler die Funktion *stunde.schuelerEvent(sID)* aufgerufen. Durch eine Abfrage der Sichtbarkeit des HTML-DIV-Elementes mit der ID *fehlzeiten* wird die Fehlzeitenerfassung gesteuert. Es wird überprüft, welcher der drei möglichen Zustände vorliegt: *anwesend, unentschuldigt abwesend* oder *verspätet*. Je nach Zustand wird das Listenelement zyklisch gewechselt:

Ein Schüler, der nicht anwesend ist, gilt zunächst als *unentschuldigt abwesend*. Erscheint der Schüler wechselt der Zustand auf *verspätet*. Durch ein erneutes Touch-Event kann der Zustand wieder auf *anwesend* gesetzt werden. Jeder Zustand wird sofort in der Datenbank gespeichert und gleichzeitig in der Übersichtsliste durch Änderung der Farbe des Schülernamens angezeigt.

4.4.3.6 Die Schülerübersicht

Durch ein Touch-Event auf ein Listenelement der linken Liste in Abbildung 50: Schülerübersicht wird die Funktion *stunde.schuelerEvent(sID)* aufgerufen. Dabei wird über die Abfrage der Sichtbarkeit des HTML-Div-Elementes mit der ID *schuelerUebersicht* die Darstellung einer Liste von Fehlzeiten, Verspätungen und Vermerken dynamisch erzeugt. Durch Auslösen eines weiteren Touch-Events auf ein Listenelement, wird die Modal-View zum Bearbeiten des Eintrages eingeblendet. Hier können die Attribute mit Freitext gefüllt, beziehungsweise über einen dynamisch erstellten Picker (SpinWheel) mit vorgegebenen Schlüsselwerten ausgewählt werden. Die eingegebenen Daten werden erst durch das Bestätigen der *Speichern* Schaltfläche in der lokalen Datenbank gespeichert oder über die Schaltfläche *Abbrechen* verworfen und die Modal View geschlossen.

4.4.3.7 Die Stundenübersicht

Die Stundenübersicht wird mit der Funktion *stunde.moveToThemaStunde()* dynamisch erstellt. Dabei wird in der rechten Ecke oben neben der Info-Schaltfläche eine Bearbeiten-Schaltfläche eingeblendet. Durch Auslösen des Touch-Events der Schaltfläche *Bearbeiten* wird die in Abbildung 53: Modal View zur Stundenübersicht beschriebe Modal View zum Bearbeiten des Eintrages eingeblendet. Hier können die Attribute *Thema, Beschreibung, Hausaufgaben zur nächsten Stunde*, sowie *Bemerkungen zu den Hausaufgaben* mit Freitext gefüllt, beziehungsweise das Attribut *Stoffverteilung* über einen dynamisch erstellten Picker (SpinWheel) mit vorgegebenen Schlüsselwerten ausgewählt werden. Die eingegebenen Daten werden erst durch das Bestätigen der *Speichern* Schaltfläche in der lokalen Datenbank gespeichert. Über die Schaltfläche *Abbrechen* wird die Eingabe hingegen verworfen und die Modal View geschlossen.

4.5 Zusammenfassung

Das Ziel der Arbeit, einen Prototyp in Form eines digitalen Klassenbuchs zu implementieren, wurde erreicht. Bei der Implementierung traten verschiedene Ablaufbesonderheiten auf, die im Wesentlichen in den noch immer nicht vollständig fehlerfreien Tools PhoneGap und DojoToolkit begründet liegen. Diese Schwierigkeiten wurden durch eigene Implementierungen oder minimale Modifikationen/Konfigurationen der Frameworks umgangen, sodass mit der vorliegenden Implementierung für iPad und Android-Systeme eine funktionsfähige Anwendung entstanden ist.

5 Ausblick und Fazit

Die Antwort auf die eingangs gestellte Frage lautet: Es ist generell möglich, native Applikationen mit einem plattformunabhängigen Ansatz zu emulieren. Dieses konnte durch die erfolgreiche Umsetzung des digitalen Klassenbuchs gezeigt werden. Entsprechend den konzeptionellen Ansätzen (Kapitel 3) und den beschriebenen Rahmenbedingungen für die Erstellung dieser Anwendung (Kapitel 4) wurde aus **einer** Codebasis je eine Applikation für iPads als auch für Android-Tablett-Systeme erstellt.

Bereits während der Entwicklung des Prototyps wurden Wünsche seitens der SfBWuG hinsichtlich einer Funktionserweiterung geäußert, die im Ausblick behandelt werden. Das abschließende Fazit fasst die Ergebnisse der gesamten Arbeit zusammen.

5.1 Ausblick

Trotz einer insgesamt zufriedenstellenden technischen Umsetzung der plattformübergreifenden Anwendung, haben sich unterschiedliche Ergänzungen als wünschenswert erwiesen, welche jedoch in ihrer Umsetzung den Rahmen dieser Arbeit gesprengt hätten. Im Folgenden werden eine Reihe möglicher Ergänzungen dargestellt.

Allgemein sollte das Layout für die verschiedenen Stile der Benutzungsoberfläche weiter verfeinert werden. Da in der nächsten Zeit die mobilen Tablett-Systeme mit Windows als Betriebssystem auf den Markt drängen, wäre es wünschenswert auch für diese Benutzungsoberfläche entsprechende CSS-Dateien im DojoToolkit zur Verfügung zu haben.

Sollte sich eine Unterstützung für Windows-Tabletts als zwingend erforderlich herausstellen und die entsprechenden CSS-Dateien im DojoToolkit nicht zur Verfügung stehen, könnte im Falle des Klassenbuches in Betracht gezogen werden, das Verhalten der einzelnen Elemente durch eine eigene Realisierung vom DojoToolkit abzukoppeln. Würde dies konsequent auch für das iOS und Android mit entsprechendem Aufwand realisiert werden, könnte damit der Einsatz einer Softwarebibliothek wie das DojoToolkit gespart werden. Dies hätte den Vorteil, dass

die Schwierigkeiten eines *custombuilds* bei Funktionserweiterungen umgangen werden.

Als funktionale Erweiterungen bei einer Weiterentwicklung des Klassenbuches sollte die Möglichkeit des Vertretungsunterrichtes vorgesehen werden. Hierzu müsste die Datenschnittstelle der SfBWuG entsprechend erweitert werden. Für die Anwendung selbst wäre es lediglich wünschenswert, die zu vertretenden Stunden farblich im Stundenplan zu kennzeichnen. Weiter besteht auch der Wunsch für die Erstellung eines digitalisierten Sitzplans Fotos aus der aktuellen Unterrichtssituation aufzunehmen und in die Darstellung aus Kapitel 14.3.1.3.1 Fehlzeiterfassung zu integrieren.

Eine wichtige Funktionalität für Klassenlehrer sollte dadurch geschaffen werden, dass diese alle Informationen ihrer Klassen angezeigt bekommen können. Dafür müsste die derzeitige Implementierung durch das Merkmal *Klassenlehrer* erweitert werden. Bisher werden nur die Eintragungen von Schülern dem jeweiligen Lehrer angezeigt, der diese vorgenommen hat.

Eine Implementierung zur Dokumentation von Unterrichtsmaterialien mit einem, idealer Weise, direkten Zugriff auf diese Materialien, wäre ebenfalls vorstellbar. Schon heute sind prinzipiell die Lehrkräfte im Zuge des Qualitätsmanagements an Bremer Schulen dazu angehalten, diese sogar an zentraler Stelle zu sammeln (vgl. SfBWuG, 2012, S. 16 u. 30).

Auch die Unterstützung der Erstellung von Zeugnissen könnte eingebunden werden. Sie erfolgt derzeit an den Bremer Schulen durch myFuNE (Fehlzeiten und Noteneingabe), eine Eigenentwicklung als Ergänzung zum Schulverwaltungsprogramm. Es bietet sich an, diese Notenerfassung in Verbindung mit weiteren diagnostischen Merkmalen bei einer Weiterentwicklung zu integrieren. Hierfür wäre es allerdings von Vorteil eine weitere, eigenständige Applikation zu erstellen, da laut Style-Guides das Überladen einer Applikation mit Funktionalitäten zu vermeiden ist.

Für die Hausaufgaben wäre es wünschenswert, einen Abgabetermin mit angeben zu können. In der aktuellen Version können aufgrund der Datenstruktur des

Schulverwaltungsprogramms lediglich die Hausaufgaben der letzten Stunde angezeigt werden.

Auf Wunsch der Senatorin für Bildung, Wissenschaft und Gesundheit wurde auf eine Style-Guide konforme Umsetzung für eine Android-Implementierung verzichtet, damit für den Support und das zu erstellende Benutzerhandbuch nur eine grafisch identische Oberfläche dokumentiert werden muss. Eine solche Umsetzung wäre eine mögliche Ergänzung dieser Arbeit.

5.2 Fazit

Um eine Benutzungsoberfläche zu entwickeln, die die Handhabung und optische Darstellung einer nativen Anwendung weitgehend umsetzt, ist das DojoToolkit ein guter Einstiegspunkt für die Anwendungsentwicklung. Allerdings muss in Kauf genommen werden, dass nicht alle nativen Elemente der Style-Guides vollständig unterstützt werden. Insbesondere ist die CSS Implementierung der Android-Oberfläche nicht vollständig abgeschlossen und auch in der Blackberry-Oberfläche fehlen einige wichtige Elemente, die für diese Plattformen nachimplementiert werden müssten.

Eine bedeutsamere Einschränkung liegt im derzeitigen Fehlen einer Implementierung des DojoToolkits für das Windows Betriebssystem Apollo/Metro. Dadurch ist es unklar, ob eine Umsetzung der spezifischen Elemente der Benutzungsoberfläche über das DojoToolkit für dieses Betriebssystem gewährleistet ist.

Bei der Entwicklung plattformunabhängiger Applikationen verlagert sich bei umfangreicheren Anforderungen, wie dem im praktischen Teil dieser Arbeit implementierten Klassenbuchs, der Aufwand auf die Erstellung der Benutzungsoberfläche. Im Gegensatz zu den nativen Gegenstücken ergeben sich gelegentlich Seiteneffekte bei der dynamischen Platzierung von Objekten, die den Aufwand dieses Teils der Entwicklung deutlich erhöhen.

Der Vollständigkeit halber muss auch erwähnt werden, dass in seltenen Fällen nicht reproduzierbare Effekte bei der Benutzungsoberfläche beobachtbar waren. Im iOS zum Beispiel rücken die Stundenpläne beim Nachladen einer neuen Woche kurzzeitig nach unten und in der Android-Version kommt es zu kleineren Grafik-Bugs.

In seltenen, nicht reproduzierbaren Situationen kann es dazu kommen, dass durch das Ändern des Listeninhaltes erst nach einem Bewegungs-Event der eigentliche Inhalt angezeigt wird und in dieser Zeit vom Listeninhalt nur der weiße Hintergrund erscheint. Diese Ablaufbesonderheiten beeinträchtigen die Benutzung zwar nicht, sollten aber im Falle eines produktiven Einsatzes dieses Prototypen im Zusammenhang mit Systemupdates und Release-Wechseln von PhoneGap oder DojoToolkit evaluiert werden.

Das Ergebnis dieser Arbeit zeigt, das Konzept der Hybrid-Anwendung, Webtechnologien und devicespezifische Schnittstellen zu kombinieren, auch für komplexere Projekte eine sinnvolle Alternative zur plattformspezifischen Implementierung darstellt.

6 Abbildungsverzeichnis

Abbildung 1: Tablett-Systeme Marktanteile Deutschland (FAZ, 2011) 6
Abbildung 2: Smartphone Plattformen mit dazugehörigen Programmiersprachen 7
Abbildung 3: Programmfluss der nativen iOS Benutzungsoberfläche (Apple Inc) 8
Abbildung 4: Aufbau des Android Betriebssystems (Google Inc., 2012) 8
Abbildung 5: Konzept für ein Plattformunabhängiges Framework 10
Abbildung 6: Aufbau des Rhodes-Framework (Rhomobile, 2011) 13
Abbildung 7: Hardwareunterstützung von mobilen Geräten (Nitobi) 15
Abbildung 8: Entwicklungswerkzeug von Application Craft (Application Craft, 2012) 16
Abbildung 9: Vergleich der nativen Elemente des iOS mit Android 21
Abbildung 10: Realisierbarkeit der nativen iOS Elemente mit dem DojoToolkit 23
Abbildung 11: Parameter der Tool Bar 26
Abbildung 12: Die iOS Tool Bar (Apple Inc., 2011, S. 100) 26
Abbildung 13: Die neue Tool Bar 26
Abbildung 14: Parameter der Popover View 28
Abbildung 15: Die iOS Popover View (Apple Inc., 2011, S. 103) 28
Abbildung 16: Die neue Popover View 28
Abbildung 17: Das iOS Action Sheet (Apple Inc., 2011, S. 120) 30
Abbildung 18: Das neue Action Sheet 30
Abbildung 19: Die iOS Modal View (Apple Inc., 2011, S. 122) 31
Abbildung 20: Die neue Modal View 31
Abbildung 21: Parameter der Modal View 31
Abbildung 22: Der iOS Activity Indicator (Apple Inc., 2011, S. 124) 32
Abbildung 23: der neue Activity Indicator 32
Abbildung 24: Der iOS Detail Disclosure Button (Apple Inc., 2011, S. 126) 33
Abbildung 25: Der neue Detail Disclosure Button 33
Abbildung 26: Parameter der Search Bar 34
Abbildung 27: Die iOS Search Bar (Apple Inc., 2011, S. 131) 35
Abbildung 28: Die neue Search Bar 35
Abbildung 29: Rate der Schwänzer, die keine Reaktion auf ihr Schwänzen erlebt haben nach Intensität des Schulschwänzens (Wetzels & Brettfeld, 2009, S. 102) 37
Abbildung 30: Institutionen die an der Erhebung und Auswertung von Daten des digitalen Klassenbuches beteiligt sind 38
Abbildung 31: Einordnung der elektronischen Fehlzeitenerfassung (EKB) in den Schulbetrieb 41
Abbildung 32: Anforderungsspezifikation der Schulen (ifib, 2011, S. 2) 44
Abbildung 33: Die Netzinfrastruktur des Schulnetzes 48
Abbildung 34: ER-Schemas für die Tabellen LOGIN OPTIONS und STDRASTER 55

6 Abbildungsverzeichnis

Abbildung 35: ER-Schema der EKB Tabelle 56
Abbildung 36: ER-Schema der Tabellen EKATEGORIEN und SFEHLKATEGORIEN 56
Abbildung 37: ER-Schema der Tabellen SFEHLGRUENDE und EGRUENDE 57
Abbildung 38: ER-Schemas der Tabellen MASSNAHMEN und EINTRAGSKATEGORIEN 57
Abbildung 39: ER-Schema der Tabelle SFEHLZEITEN 58
Abbildung 40: ER-Schema der Tabelle SVERMERKE 59
Abbildung 41: ER-Schema der Tabelle THEMASTUNDEN 60
Abbildung 42: Die Tabelle Stoffverteilgung 60
Abbildung 43: ER-Schema der Tabelle HAUSAUFGABEN 61
Abbildung 44: ER-Schema der Tabelle PICS 61
Abbildung 45: ER-Übersicht der Relationen in Min-Max-Notation 62
Abbildung 46: Programmfluss des elektronischen Klassenbuchs 64
Abbildung 47: Login Screen auf einem Android System 65
Abbildung 48: Stundenplan Screen auf einem Android System 66
Abbildung 49: Fehlzeitenerfassung 67
Abbildung 50: Schülerübersicht 68
Abbildung 51: Stundenübersicht 69
Abbildung 52: Modal View zum Lern und Sozialverhalten 70
Abbildung 53: Modal View zur Stundenübersicht 70
Abbildung 54: PopoverView: Optionen 70
Abbildung 55: PopoverView: Hilfe 70

7 Literaturverzeichnis

Apple Inc. (01. 10 2011). Deploying iPhone and iPad Exchange ActiveSync. Cupertino, Californien, USA: Apple Inc.

Apple Inc. (23. 03 2011). iOS Human Interface Guidelines. (2011-03-23). Cupertino, Kalifornien, USA.

Breiter, A. (2006). Mediennutzung in Bremer Schulen Studie zum Einsatz digitaler Medien in stadtbremischen Schulen aus Sicht von Schulen, Lehrkräften und Schülerinnen und Schülern. ifib. Bremen: ifib.

Hövelmann, W. (10. 02 2011). Prüfung der Anweneheitslisten aus dem Klassenbuch. (R.-P. Hinze, Interviewer)

Hansen, M. (01. 09 2011). Befragung zum Projekt elektronische Erfassung von Fehlzeiten. (B. Hinze, Interviewer) Bremen, Bremen, Deutschland: SfB.

ifib. (2011). Wissenschaftliche Unterstützung bei der Evaluation und Einführung eines elektronischen Klassenbuches an fünf bremer Pilotschulen. Bremen: ifib.

Kemper, A., & Eickler, A. (2006). *Datenbanksysteme : eine Einführung* (6th ed.). München: Oldenbourg.

Russell, M. A. (2008). *Dojo: The Definitive Guide.* 2008: O'Reilly Media.

SfBW. (2002). Deputationsvorlage G189. Bremen: SfBW.

SfBW. (1998). Verordnung über das Verfahren beim erlass von Ordnungsmaßnahmen in der Schule. Brmen: SfBWuG.

SfBWuG. (2012). Richtlinien über die Verarbeitung von personenbezogenen Daten und zur Führung von Schullaufbahnakten in der Stadtgemeinde Bremen. SfBWuG. Bremen: SfBWuG.

Wetzels, P., & Brettfeld, K. (2009). *Gewalt und Delinquenz junger Menschen in Bremen 2008 – 2010.* Fakultät für Rechtswissenschaften. Hamburg: Universität Hamburg.

Onlinequellen:

Apers, C. (06. 02 2010). *WebApp.Net*. Abgerufen am 02. 12 2011 von http://webapp-net.com/

Appcelerator Inc. (01. 08 2010). Mobile app platform for building mobile apps from web technologies for Android, iOS, iPhone, iPad and mobile web. Abgerufen am 13. 04 2012 von http://www.appcelerator.com/

Apple Inc. (kein Datum). *Cocoa Touch*. Abgerufen am 12. 04 2012 von https://developer.apple.com/technologies/ios/cocoa-touch.html

Application Craft. (06. 01 2012). *Application Craft developing a better way*. Abgerufen am 13. 04 2012 von http://www.applicationcraft.com/

Beck-online die Wissensdatenbank. (10. 03 2007). *BremSchulG §3 Datenzugang und Nutzung außerschulischer Datenverarbeitungsgeräte - beck-online*. Abgerufen

am 02. 12 2011 von http://beck-online.beck.de/?vpath=bibdata%2Fges%2FBrSchulDSG%2Fcont%2FBrSchulDSG%2EP3%2Ehtm

Beck-online Die Wissensdatenbank. (11. 08 2009). *BremSchulG: §3 Allgemeines - beck-online*. Abgerufen am 02. 12 2011 von http://beck-online.beck.de/default.aspx?vpath=bibdata%2fges%2fBrSchulG%2fcont%2fBrSchulG.P3.htm

Dalu, M. (13. 01 2010). *Mobile Apps cross-platform development challenge: PhoneGap vs. Titanium vs. Rhodes*. Abgerufen am 13. 04 2012 von http://surgeworks.com/blog/lab-mobile/iphone/mobile-apps-cross-platform-development-challenge-phonegap-vs-titanium-vs-rhodes

doughays, r. (28. 10 2011). *Opener behavior very strange on android phone*. Abgerufen am 13. 04 2012 von http://bugs.dojotoolkit.org/ticket/14165

Falk, K. (01. 03 2012). *Mobile Frameworks Comparison Chart*. Abgerufen am 12. 04 2012 von http://www.markus-falk.com/mobile-frameworks-comparison-chart/

FAZ. (06. 12 2011). *F.A.Z. Community*. Abgerufen am 06. 12 2011 von http://193.227.146.30/blogs/netzkonom/archive/2011/09/15/apples-ipad-nimmt-den-android-tablets-wieder-marktanteile-ab.aspx

Götz, K. (02. 12 2011). *Senatorin für Bildung und Wissenschaft - Inklusion*. Abgerufen am 02. 12 2011 von http://bildung.bremen.de/sixcms/detail.php?gsid=bremen117.c.24554.de

Google Inc. (20. 10 2011). *Android Design*. Abgerufen am 20. 01 2012 von http://developer.android.com/design/index.html

Google Inc. (07. 04 2012). *What is Android?* Abgerufen am 12. 04 2012 von http://developer.android.com/guide/basics/what-is-android.html

Gundersen, M. (kein Datum). *touch*. Abgerufen am 13. 04 2012 von http://lab.mariusgundersen.com/touch.html

Keese, B. (17. 06 2011). *[Dojo-interest] Is dojo.require a problem on android devices?* Abgerufen am 15. 04 2012 von http://mail.dojotoolkit.org/pipermail/dojo-interest/2011-June/055794.html

Microsoft. (2012). *UX-Richtlinien für Apps im Metro-Stil*. Abgerufen am 13. 04 2012 von http://msdn.microsoft.com/library/windows/apps/hh465424

Nitobi. (kein Datum). *PhoneGap*. Abgerufen am 21. 02 2012 von www.phonegap.com

Nitobi. (kein Datum). *Supported Features - PhoneGap*. Von http://phonegap.com/about/features abgerufen

Rhomobile. (6. 07 2011). *Rhoemobile*. Abgerufen am 12. 04 2012 von http://docs.rhomobile.com/rhodes/introduction

Simpson, K. (30. 10 2011). *JSON-P: Safer cross-domain Ajax with JSON-P/JSONP*. Abgerufen am 13. 06 2012 von http://www.json-p.org/

Symantec. (kein Datum). *Mobile Device Management | Symantec*. Abgerufen am 13. 04 2012 von http://www.symantec.com/mobile-device-management

The Dojo Foundation. (16. 02 2012). *DojoToolkit*. Abgerufen am 13. 04 2012 von http://dojotoolkit.org/

The Dojo Foundation. (01. 12 2011). *The DojoToolkit Blog*. Abgerufen am 13. 04 2012 von http://dojotoolkit.org/blog/dojo-1-7-released

Z., J. (21. 09 2010). *Loading spinner animation using CSS and WebKit*. Abgerufen am 13. 04 2012 von http://37signals.com/svn/posts/2577-loading-spinner-animation-using-css-and-webkit

7 Literaturverzeichnis

Apple Inc. (01. 10 2011). Deploying iPhone and iPad Exchange ActiveSync. Cupertino, Californien, USA: Apple Inc.

Apple Inc. (23. 03 2011). iOS Human Interface Guidelines. (2011-03-23). Cupertino, Kalifornien, USA.

Breiter, A. (2006). Mediennutzung in Bremer Schulen Studie zum Einsatz digitaler Medien in stadtbremischen Schulen aus Sicht von Schulen, Lehrkräften und Schülerinnen und Schülern. ifib. Bremen: ifib.

Hövelmann, W. (10. 02 2011). Prüfung der Anweneheitslisten aus dem Klassenbuch. (R.-P. Hinze, Interviewer)

Hansen, M. (01. 09 2011). Befragung zum Projekt elektronische Erfassung von Fehlzeiten. (B. Hinze, Interviewer) Bremen, Bremen, Deutschland: SfB.

ifib. (2011). Wissenschaftliche Unterstützung bei der Evaluation und Einführung eines elektronischen Klassenbuches an fünf bremer Pilotschulen. Bremen: ifib.

Kemper, A., & Eickler, A. (2006). *Datenbanksysteme : eine Einführung* (6th ed.). München: Oldenbourg.

Russell, M. A. (2008). *Dojo: The Definitive Guide.* 2008: O'Reilly Media.

SfBW. (2002). Deputationsvorlage G189. Bremen: SfBW.

SfBW. (1998). Verordnung über das Verfahren beim erlass von Ordnungsmaßnahmen in der Schule. Brmen: SfBWuG.

SfBWuG. (2012). Richtlinien über die Verarbeitung von personenbezogenen Daten und zur Führung von Schullaufbahnakten in der Stadtgemeinde Bremen. SfBWuG. Bremen: SfBWuG.

Wetzels, P., & Brettfeld, K. (2009). *Gewalt und Delinquenz junger Menschen in Bremen 2008 – 2010.* Fakultät für Rechtswissenschaften. Hamburg: Universität Hamburg.

Onlinequellen:

Apers, C. (06. 02 2010). *WebApp.Net*. Abgerufen am 02. 12 2011 von http://webapp-net.com/

Appcelerator Inc. (01. 08 2010). Mobile app platform for building mobile apps from web technologies for Android, iOS, iPhone, iPad and mobile web. Abgerufen am 13. 04 2012 von http://www.appcelerator.com/

Apple Inc. (kein Datum). *Cocoa Touch*. Abgerufen am 12. 04 2012 von https://developer.apple.com/technologies/ios/cocoa-touch.html

Application Craft. (06. 01 2012). *Application Craft developing a better way*. Abgerufen am 13. 04 2012 von http://www.applicationcraft.com/

Beck-online die Wissensdatenbank. (10. 03 2007). *BremSchulG §3 Datenzugang und Nutzung außerschulischer Datenverarbeitungsgeräte - beck-online*. Abgerufen am 02. 12 2011 von http://beck-online.beck.de/?vpath=bibdata%2Fges%2FBrSchulDSG%2Fcont%2FBrSchulDSG%2EP3%2Ehtm

Beck-online Die Wissensdatenbank. (11. 08 2009). *BremSchulG: §3 Allgemeines - beck-online*. Abgerufen am 02. 12 2011 von http://beck-online.beck.de/default.aspx?vpath=bibdata%2fges%2fBrSchulG%2fcont%2fBrSchulG.P3.htm

Dalu, M. (13. 01 2010). *Mobile Apps cross-platform development challenge: PhoneGap vs. Titanium vs. Rhodes*. Abgerufen am 13. 04 2012 von http://surgeworks.com/blog/lab-mobile/iphone/mobile-apps-cross-platform-development-challenge-phonegap-vs-titanium-vs-rhodes

doughays, r. (28. 10 2011). *Opener behavior very strange on android phone*. Abgerufen am 13. 04 2012 von http://bugs.dojotoolkit.org/ticket/14165

Falk, K. (01. 03 2012). *Mobile Frameworks Comparison Chart*. Abgerufen am 12. 04 2012 von http://www.markus-falk.com/mobile-frameworks-comparison-chart/

FAZ. (06. 12 2011). *F.A.Z. Community*. Abgerufen am 06. 12 2011 von http://193.227.146.30/blogs/netzkonom/archive/2011/09/15/apples-ipad-nimmt-den-android-tablets-wieder-marktanteile-ab.aspx

Götz, K. (02. 12 2011). *Senatorin für Bildung und Wissenschaft - Inklusion*. Abgerufen am 02. 12 2011 von http://bildung.bremen.de/sixcms/detail.php?gsid=bremen117.c.24554.de

Google Inc. (20. 10 2011). *Android Design*. Abgerufen am 20. 01 2012 von http://developer.android.com/design/index.html

Google Inc. (07. 04 2012). *What is Android?* Abgerufen am 12. 04 2012 von http://developer.android.com/guide/basics/what-is-android.html

Gundersen, M. (kein Datum). *touch*. Abgerufen am 13. 04 2012 von http://lab.mariusgundersen.com/touch.html

Keese, B. (17. 06 2011). *[Dojo-interest] Is dojo.require a problem on android devices?* Abgerufen am 15. 04 2012 von http://mail.dojotoolkit.org/pipermail/dojo-interest/2011-June/055794.html

Microsoft. (2012). *UX-Richtlinien für Apps im Metro-Stil*. Abgerufen am 13. 04 2012 von http://msdn.microsoft.com/library/windows/apps/hh465424

Nitobi. (kein Datum). *PhoneGap*. Abgerufen am 21. 02 2012 von www.phonegap.com

Nitobi. (kein Datum). *Supported Features - PhoneGap*. Von http://phonegap.com/about/features abgerufen

Rhomobile. (6. 07 2011). *Rhoemobile*. Abgerufen am 12. 04 2012 von http://docs.rhomobile.com/rhodes/introduction

Simpson, K. (30. 10 2011). *JSON-P: Safer cross-domain Ajax with JSON-P/JSONP*. Abgerufen am 13. 06 2012 von http://www.json-p.org/

Symantec. (kein Datum). *Mobile Device Management | Symantec*. Abgerufen am 13. 04 2012 von http://www.symantec.com/mobile-device-management

The Dojo Foundation. (16. 02 2012). *DojoToolkit*. Abgerufen am 13. 04 2012 von http://dojotoolkit.org/

The Dojo Foundation. (01. 12 2011). *The DojoToolkit Blog*. Abgerufen am 13. 04 2012 von http://dojotoolkit.org/blog/dojo-1-7-released

Z., J. (21. 09 2010). *Loading spinner animation using CSS and WebKit*. Abgerufen am 13. 04 2012 von http://37signals.com/svn/posts/2577-loading-spinner-animation-using-css-and-webkit