Mobile Webanwendungen mit jQuery Mobile und PrimeFaces Mobile

FWP: Aktuelle Technologien zur Entwicklung verteilter Java Anwendungen

Version: 1.0
Datum: 09.05.2013

Name: Giebelhaus Alexander
Matrikelnummer: 05702310
Semester: Sommersemester 2013 – Wirtschaftsinformatik IB6A

Ich erkläre hiermit, dass ich die vorliegende Studienarbeit selbständig verfasst, diese Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt, dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und dass ich wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

_____ _____

Ort, Datum Unterschrift Student

Inhaltsverzeichnis

1 Die mobile Welt

Die Nutzung des Internets mit mobilen Endgeräten ist heutzutage Normalität. Laut einer Statistik von Accenture waren 2012 im Durchschnitt 56% der weiblichen und 73% der männlichen Internetnutzer weltweit mit einem mobilen Endgerät online. In Deutschland sind es immerhin 58% der Frauen und 59% der Männer. Das heißt das mittlerweile gut die Hälfte der Internetnutzer ein mobiles Gerät bevorzugen.

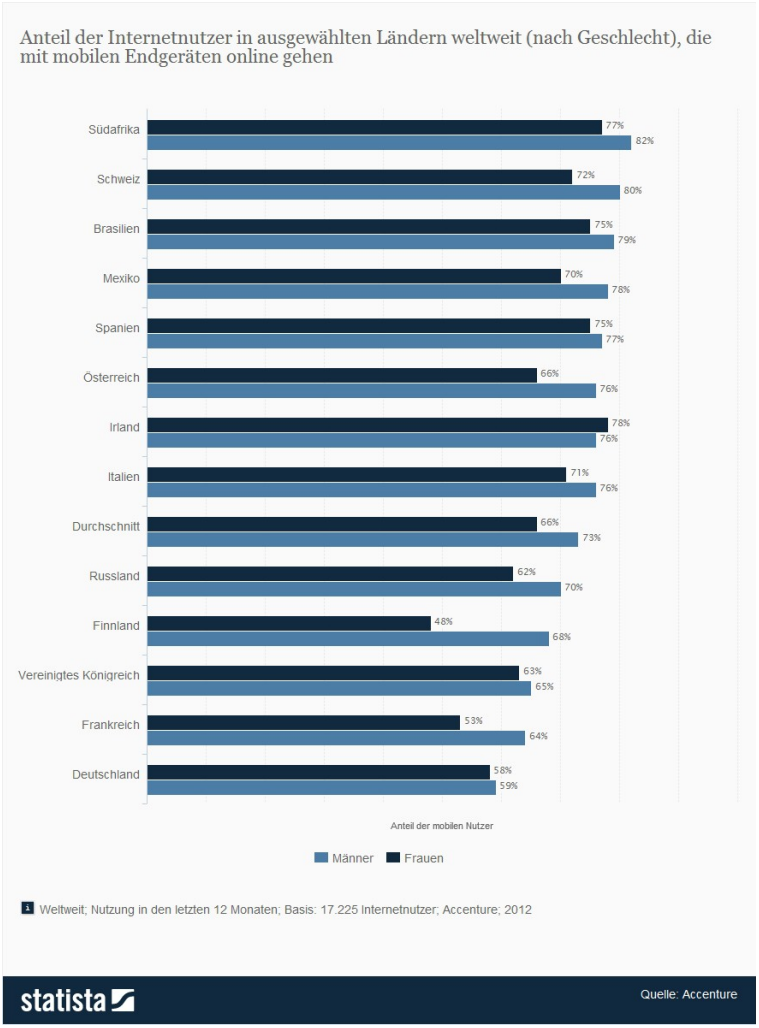


Abbildung 1: Mobile Internetnutzung (http://de.statista.com/)

1.1 Klassifizierung mobiler Geräte

Die Leute nutzen das Internet verstärkt über mobile Geräte. Heutzutage haben wir unterschiedliche Geräte mit verschiedenen Bildschirmgrößen, Eingabemöglichkeiten und neuen

Technologien. Es gibt mittlerweile mehr als fünf Millionen dieser Geräte. Die Geräte lassen sich dabei lose in folgende Klassen unterteilen[1]:

Mobiltelefone

Diese werden hauptsächlich zum Telefonieren und SMS schreiben genutzt. Diese haben keine Internetverbindung und nutzen das Web nicht. Sie sind für uns nicht relevant.

Low-end Mobilgeräte

Low-end Mobilgeräte haben bereits einen Browser um im Internet zu surfen. Jedoch ist dieser nur mit Basisfunktionen ausgestattet. Die Geräte haben ebenfalls ein sehr kleines Display und eine begrenzte Leistung.

Mid- and high end Mobilgeräte

Diese Geräte haben bereits einen HTML-Browser und eine mittlere Bildschirmgröße. Sie sind bereits touchfähig, haben eine gute Kamera, Musikplayer und Spiele. Zur Verbindung mit dem Internet wird hauptsächlich das W-LAN verwendet.

Smartphones

Ein Smartphone zeichnet sich dadurch aus, dass es oft mit einer Flatrate angeboten wird. Das Betriebssystem ist Multitaskingfähig, hat einen modernen HTML5 Browser, unterstützt WLAN und 3G hat einen Musikplayer sowie meistens GPS, Bluetooth, ein touchfähiges Display, eine videofähige Kamera und man kann Anwendungen (Apps) installieren.

Tablets

Tablets haben einen größeren Bildschirm, haben einen HTML5-Browser, WLAN und manchmal 3G. Sie haben ebenfalls Touch und viele andere Funktionen von Smartphones. Vorreiter ist hier das iPad von Apple. Mittlerweile existieren aber auch andere Geräte von anderen Herstellern.

1.2 Nutzung der mobilen Endgeräte

Als mobile Endgeräte werden insbesondere Smartphones, Tablets, mobile Spielkonsolen (PSP), MP3-Player (iPod) sowie E-Reader verwendet. Doch was bewegt diese Leute dazu unterwegs im Internet zu surfen. Es ist nicht verwunderlich, dass die meisten nach Informationen suchen, wie es auch bei der herkömmlichen Internetnutzung der Fall ist. Ganze 87% wollen sich einfach auf dem Laufenden halten. Da die Nutzer mit mobilen Endgeräten nicht an das Arbeitszimmer gebunden sind, können Informationen von überall abgerufen werden. Es werden dadurch Informationen

1 vgl. (Firtman, Maximiliano R. 2012, http://proquest.tech.safaribooksonline.de/book/-/9781449331085/the-mobile-and-tablet-world/id2595201)

welche sich auf den aktuellen Ort beziehen abgefragt, wie z.B. „Welche Restaurants sind in der Nähe?" oder „wo finde ich die nächste Bushaltestelle?". Diese Art von Services verwenden 68%. Die anderen nutzen das Internet unterwegs hauptsächlich zur Zeitersparnis, zum Einkauf, zur Unterhaltung oder gegen die Langeweile.

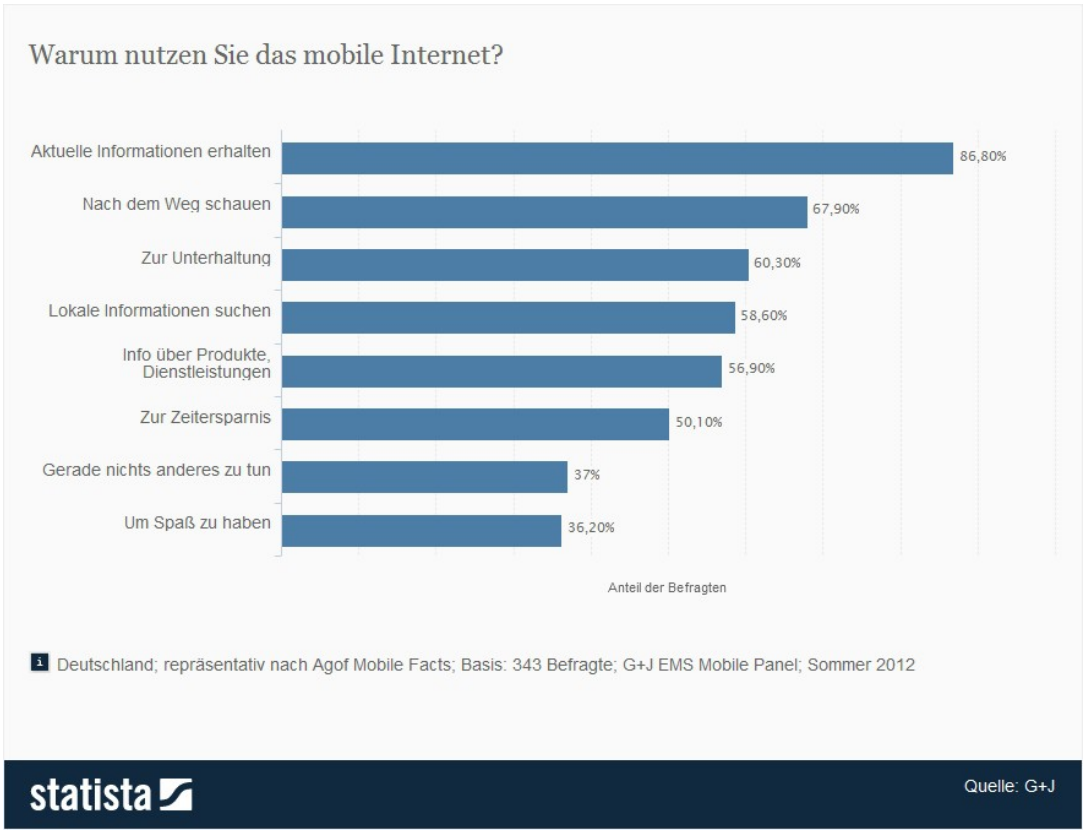


Abbildung 2: Warum nutzen Sie das mobile Internet? (http://de.statista.com/)

Deswegen ist es mittlerweile unabdingbar, für diese Endgeräte die Inhalte entsprechend angepasst auszuliefern. Es ist zwar mit den Geräten auch möglich normale Webseiten aufzurufen, diese sind jedoch oft schwerfällig zu bedienen. Gerne wird in dieser Verbindung vom „mobilen Web" gesprochen, dieser Begriff leitet jedoch zu der Annahme, dass es ein eigenes Internet nur für die mobilen Geräte ist – dies ist jedoch nicht der Fall.

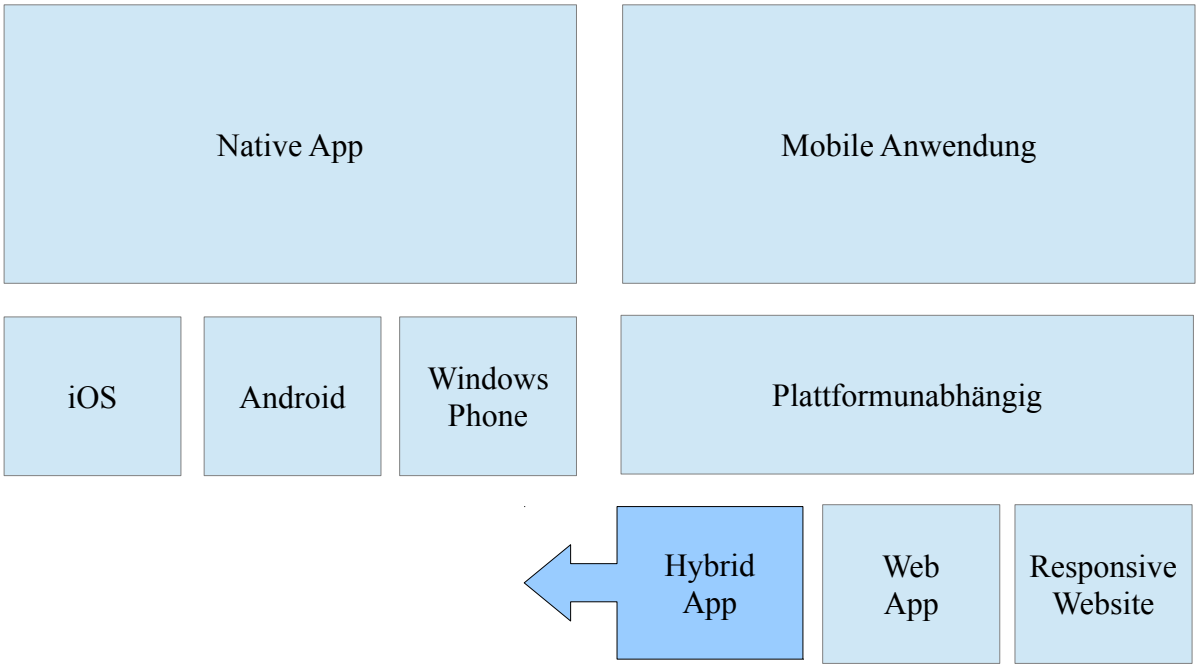
1.3 Native App vs. Mobile Anwendung

Wichtig ist es den Unterschied zwischen nativen Apps und mobilen Webanwendungen zu verstehen. Vor der Realisierung einer Applikation auf mobilen Plattformen muss man sich entscheiden welchen Weg man dabei gehen will. Native Apps haben den Vorteil, dass diese auf dem jeweiligen Gerät mit vollem Zugriff auf die Hardware ablaufen. Mobile Webseiten werden mit einem Browser auf dem mobilen Gerät aufgerufen und haben nicht auf alle Hardwarekomponenten

Zugriff.

Warum sollte man dann überhaupt mobile Webanwendungen erstellen?
Der Hauptvorteil liegt dabei in der Plattformunabhängigkeit, bei nativen Applikationen muss diese entsprechend für die gängigen Systeme IOS, Android, WindowsPhone programmiert werden. Dabei kommen hier schon alleine drei unterschiedliche Programmiersprachen zum Einsatz (Java, C#, Objective-C), was die Programmierung für alle Plattformen sehr kostenintensiv macht. Bei mobilen Webanwendungen kann die Anwendung über alle Plattformen aufgerufen werden.

Es gibt mittlerweile auch Möglichkeiten mobile Webanwendungen als native Apps laufen zu lassen (sog. hybrid).



Zeichnung 1: Native vs. Mobile

2 UI-Design für mobile Endgeräte

„User interface design or user interface engineering is the design of computers, appliances, machines, mobile communication devices, software applications, and websites ***with the focus on the user's experience and interaction****."* [2]

Dies bedeutet für uns, dass der Nutzer bei der Konzeption unserer Anwendung im Vordergrund stehen sollte. Die Ziele sollten auf die Bedürfnisse des Nutzers ausgelegt werden. Der Fokus sollte dabei auf dem Inhalt liegen und ein gewisses Nutzungserlebnis bieten. „L*ess is more*" könnte man hier auch sagen.

2.1 Bedienung und Interaktion

Mit den mobilen Devices sind auch neue Eingabemöglichkeiten hinzugekommen. Dazu gehört die Benutzung per Touch (scrollen, zoomen, wischen). Durch die kleine Tastatur (welche auch nur bei Bedarf eingeblendet werden kann) sind Texteingaben sehr mühsam und dauern länger als mit den „alten" Tastaturen. Diese und weitere Aspekte müssen bei dem Design des User-Interfaces beachtet werden. Durch das „anywhere-everywhere"-Prinzip können Inhalte von überall abgerufen werden, wodurch die Nutzer weniger fokussiert und schneller abgelenkt sind. Deswegen müssen die Informationen kurz und knackig präsentiert und auf das Wesentliche komprimiert werden.

Das User-Interface muss den User dabei unterstützen schnell und effizient an sein Ziel zu kommen:[3]

- Auf der Startseite sollten die wichtigsten Punkte direkt aufrufbar sein. Dies wird oft als Liste realisiert, welche auf die wichtigsten Features bzw. Inhalte aus Sicht der Nutzer verlinken (meistens in vertikaler Anordnung).
- Die wichtigsten Links sollten zuerst aufgeführt sein.
- Durch die Bedienung mit Finger oder Stift, müssen die Bedienelemente eine gewisse Größe aufweisen um eindeutig getroffen werden zu können.
- Beim Auslösen von Events sollte ein entsprechendes Feedback zurückkommen z.B. der Button ändert bei Berührung seine Farbe.
- Durch den geringen Platz sollte die Navigation per Liste stattfinden und mit Zurück-

2 http://en.wikipedia.org/wiki/User_interface_design

3 http://mobile.smashingmagazine.com/2011/05/02/a-user-centered-approach-to-mobile-design/

Buttons gearbeitet werden.

- Da die mobile Anwendung meistens nur einen Kernbereich der Webseite darstellt, sollte man im unteren Bereich der Seite dem User immer auch den Zugriff auf die Desktop-Version ermöglichen (zumindest bei mobilen Webseiten).

Durch die bereits erwähnten Probleme bei der Texteingabe sollten Formulare wirklich nur die wichtigsten Daten vom User abfragen. Es bietet sich an diese mit bereits bekannten Werten zu belegen oder aus dem Kontext zu erschließen. Dazu gehören unter anderem:[4]

- Ortsangaben per GPS in Erfahrung bringen.
- „Auto-complete" bei Texteingaben (z.B. bei „Mü" erscheint „München" zur Auswahl).
- Felder vor füllen mit alten Werten, aktuellem Datum oder Erfahrungswerten.
- Captchas wo immer möglich deaktivieren.
- Andere Eingabemöglichkeiten falls möglich mit einbeziehen (z.B. Sprache).
- Sich die Logindaten zu merken.
- Auswahlfelder sind immer besser als Textfelder.

Außerdem kann durch die Wahl einer kurzen Webadresse die Texteingabe durch den User minimiert werden. Alternativ kann direkt bei Zugriff auf die Desktop-Version auf die mobile Version weitergeleitet werden.

2.2 Mobile Besonderheiten

Mobile Webanwendungen sind an bestimmte Restriktionen gebunden:

- Der Nutzer kann das Netz (die Verbindung) verlieren oder eine schlechte Verbindung haben.
- Die Datenrate und das Datenvolumen sind begrenzt.
- Das Display bietet einen begrenzten Platz und entsprechend geringen Kontrast (Sonneneinstrahlung).
- Das Gerät kann im Hoch- oder Querformat gehalten werden.
- Browser haben gerätespezifische Anpassungen.

4 http://mobile.smashingmagazine.com/2012/07/12/elements-mobile-user-experience/

Da der Nutzer ein schlechtes oder kein Netz haben kann, sollte dem User beim Laden von Inhalten entsprechend ein Ladesymbol angezeigt werden. So weiß dieser, dass die Anwendung noch Daten laden muss. Durch eine begrenzte Datenrate und ein eventuell begrenztes Datenvolumen durch den Netzanbieter sollten die Dateigrößen klein gehalten werden. Dies kann unter anderem erreicht werden durch:

- Gzip-Komprimierung der Webinhalte auf der Serverseite.
- Minimierung der HTML, CSS und Javascript-Dateien.
- Grafiken optimieren.
- CSS-Sprites einsetzen (dabei werden viele kleine Grafiken in eine Grafik gepackt, um die Anzahl der Requests zu minimieren).

Die neuen Smartphones erkennen mittlerweile auch wie sie gehalten werden. Wird das Gerät gedreht, dreht sich entsprechend auch der Browser. Dadurch entsteht in der Breite neuer Platz welcher aber in der Höhe fehlt. Die Anwendung sollte für diesen Fall seine Darstellung anpassen (responsive Design).

Da jeder Hersteller sein eigenes Süppchen kocht, sehen Inhalte auf unterschiedlichen Geräten unterschiedlich aus. Deswegen ist es wichtig Webanwendungen auf mehreren Geräten zu testen.

Um einige dieser Probleme in den Griff zu bekommen, gibt es ein hervorragendes Framework welche uns bei der Erstellung von mobilen Anwendungen unterstützt. Was das Framework kann und wie man es verwendet finden Sie auf den nächsten Seiten.

3 jQuery Mobile (jQM)

3.1 Was ist jQuery Mobile?

jQuery-Mobile ist eine auf HTML5, CSS und Javascript basierende UI-Bibliothek welche auf allen populären mobilen Plattformen läuft[5]. Es basiert auf dem stabilen Javascript Framework jQuery und jQuery UI - einer Bibliothek zur Erstellung von Webanwendungen.

Das Framework liefert uns das mobile Erlebnis, welches die Nutzer von mobilen Anwendungen erwarten.

3.2 Was ist es nicht?[6]

Es ist keine mobile Alternative für das bekannten jQuery. Es basiert auf jQuery, stellt aber keinen Ersatz dafür dar.

JQM ist kein SDK für Webanwendungen. Es ist möglich das bekannte mobile Erlebnis zu garantieren jedoch kann nicht auf alle Services des Betriebssystems zugegriffen werden.

Es können nicht alle Arten von Webanwendungen damit realisiert werden, insbesondere sehr leistungsintensive Anwendungen sollten besser als native App erstellt werden.

3.3 Wie unterstützt es uns?

Wie bereits erwähnt, muss die Webanwendung dem Nutzer eine gewisse User Experience bieten. jQuery-Mobile ermöglicht genau dieses Erlebnis, indem es uns vorgefertigte UI-Komponenten zur Verfügung stellt.

JQM unterstützt Touch-Events welche bei Berührung ausgelöst werden. Außerdem nutzt es intensiv die Ajax-Technologie. Durch die Verwendung von Ajax werden Bilder und Zusatzelemente asynchron geladen, dadurch wird die Anwendung nicht blockiert. JQM ist außerdem ein sehr flexibles Framework. Das Design kann einfach über mitgelieferte Tools angepasst, Zusatzfunktionalitäten als eigenes Widgets realisiert und eingebunden werden. Damit lässt sich die mobile Webanwendung einfach an das Branding des Unternehmens anpassen. Es ist außerdem einfach zu erlernen und um eine einfache Anwendung zu realisieren muss man kein Javascript beherrschen.

5 http://www.jquerymobile.com

6 vgl. (Firtman, Maximiliano R. http://proquest.tech.safaribooksonline.de/book/-/9781449331085/the-mobile-and-tablet-world/id2595201#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODE0NDkzMzEwODUlMkZpZDI1OTUwMjcmcXVlcnk9)

3.4 Ein Ring sie alle zu knechten

jQuery Mobile setzt auf sogenanntes progressive Enhancement, indem ausgehend von einer Basisversion der Seite jedem User Agent automatisch nur die Features zusätzlich angeboten werden, die dieser auch unterstützt. Dabei werden die Geräte in verschiedene Kategorien eingeteilt. Die Geräte der Kategorie A haben das volle Nutzererlebnis mit Ajax-basierten und animierten Seitenübergängen. Die Geräte der Kategorie B nutzen keine Ajax-Technologien für die Navigation. Die Kategorie C stellt nur die reine HTML-Seite ohne Effekte dar - diese können aber ohne Probleme benutzt werden.

Platform	Version	Native	Opera Mobile				Opera Mini		Fennec		Ozone	Netfront	Phonegap
			8.5	8.65	9.5	10.0	4.0	5.0	1.0	*1.1*	0.9	4.0	0.9
iOS	v2.2.1	B											A
	v3.1.3, v3.2	A						A					A
	v4.0	A						A					A
Symbian S60	v3.1, v3.2	C	C	C		B	C	B			C	C	
	v5.0	A	C	C		A	C	A					A
Symbian UIQ	v3.0, v3.1			C							C		
	v3.2				C						C		
Symbian Platform	*3.0*	A											
BlackBerry OS	v4.5	C					C	C					
	v4.6, v4.7	C					C	B					C
	v5.0	B					C	A					A
	v6.0	A						A					A
Android	v1.5, v1.6	A											A
	v2.1	A											A
	v2.2	A				*A*		*C*		*A*			A
Windows Mobile	v6.1	C	C	C	C	B	C	B				C	
	v6.5.1	C	C	C	A	A	C	A					
	v7.0	A				A	C	A					
webOS	1.4.1	A											A
bada	1.0	A											
Maemo	5.0	B				B			C	*B*			
MeeGo	*1.1*	*A*				*A*				*A*			

Abbildung 3: Mobile graded browser support (http://jquerymobile.com/gbs/)

3.5 Erste Schritte mit jQuery Mobile

Um jQuery-Mobile in eigenen Projekten verwenden zu können muss es erst auf der Webseite (www.jquerymobile.com) heruntergeladen werden. Dazu gibt es eine minimierte sowie eine unkomprimierte Version – es sollte in produktiven Anwendungen immer die minimierte eingebunden werden. Als neues Zusatzfeature kann auch ein eigenes Package erstellt werden. Dazu gibt es ein sogenanntes „builder tool", mit diesem ist es möglich, nicht benötigte Komponenten auszuschließen und somit die Größe der Quelldateien zu minimieren. Außerdem wird jQuery selbst benötigt, welches unter www.jquery.com zu finden ist. Alternativ kann man sich anstatt jQuery Mobile lokal auf den eigenen Rechner zu laden, auch im CDN-Repository von jquery.com bedienen.

Das Framework kann nun der eigenen Seite hinzugefügt werden.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <title>Page Title</title>
        <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css" />
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.3.1/jquery.mobile-
1.3.1.min.js"></script>
    </head>
<body>
</body>
</html>
```

Der Body kann im Prinzip eine beliebige Anzahl von jQuery Mobile-Seiten enthalten. Diese sind durch div-Container begrenzt, die das *data-role-Attribut* mit dem Wert *„page"* besitzen.

Wie in dem Beispiel unten ersichtlich wird, werden außerdem weitere Komponenten eingebunden. Bei *data-role="header"* und *data-role="footer"* handelt es sich um Toolbar-Komponenten. Mit dem Attribut *data-role="content"* wird der Platz für den Inhalt der Anwendung definiert.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <title>Page Title</title>
        <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css" />
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.3.1/jquery.mobile-
1.3.1.min.js"></script>
    </head>
<body>
    <div data-role="page">
        <div data-role="header">
            <h1>Seitentitel</h1>
        </div>
        <div data-role="content">
            Inhalt
        </div>
        <div data-role="footer">
            Footer
        </div>
    </div>
</body>
</html>
```

Aus ein paar Zeilen wurde so nun unsere erste jQuery-Mobile-Anwendung.

3.6 Pages

Es gibt mehrere Wege die eigene Anwendung zu strukturieren. Es können mehrere Pages in ein Dokument gepackt werden oder für jede Page ein eigenes Dokument angelegt werden.

Im ersten Fall müssen die Pages individuell durch eine ID gekennzeichnet werden. Über diese ID funktioniert auch die Verlinkung, damit sich zwischen beiden Seiten wechseln lässt.

```
...
<body class="container">
    <div data-role="page" id="page-one" data-title="Seite 1" >
        <div data-role="header">
            <h1>Seite 1</h1>
        </div>
        <div data-role="content">Hier steht der Inhalt</div>
        <div data-role="footer">Footer</div>
    </div>
    <div data-role="page" id="page-two" data-title="Seite 2" >
        <div data-role="header">
            <h1>Seite 2</h1>
        </div>
        <div data-role="content">Hier steht der Inhalt von Seite 2</div>
        <div data-role="footer">Footer</div>
    </div>
</body>
...
```

3.6.1 Navigation

Für die Navigation in einer Anwendung gibt es mehrere Ansätze. Eine einfache Navigation auf eine zweite Page im selben Dokument funktioniert über:

```
<a href="#page-two">Seite 2</a>
```

Es empfiehlt sich jedoch, je Page eine eigenes HTML-Dokument zu erstellen. Hier wird dann einfach per Hyperlink auf das andere Dokument verlinkt. Der Inhalt wird von jQuery Mobile automatisch per Ajax geladen. Dabei wird intern ein XMLHttpRequest abgesetzt.

Das Laden per Ajax kann aber per Annotation deaktiviert werden. Verlinkungen auf folgende Art und Weise werden nicht per Ajax in das aktuelle Dokument geladen.

- Links auf eine externe Domainß (same origin policy),
- Links mit dem Attribut *rel="external"* sowie
- mit dem Attribut *data-ajax="false".*

Außerdem gibt es noch Links welche eine besondere Aktion auslösen:

- *...href="**tel:0800123123123**"...* löst einen Anruf aus,

- *...href="**mailto:me@mail.com**"...* öffnet das eigene Email-Programm mit vorgefüllter Email-Adresse.

Ein weiterer interessanter Nebeneffekt. Beim Laden von Pages zeigt das Framework mit einer Grafik an, dass Inhalte noch geladen werden. Der Benutzer weiß somit, dass die Anwendung gerade die Inhalte lädt und sich nicht verabschiedet hat. Sollte die Page dennoch nicht geladen werden können, wird eine entsprechende Nachricht an den User gesendet.

3.6.2 Seitenübergänge

Seitenübergänge geben der Webanwendung einen nativen Touch. Allein durch das Hinzufügen eines Attributes an einen Link entsteht eine tolle user-experience. Die möglichen Effekte werden durch das *data-transition*-Attribut gesteuert.

Folgende Werte sind für *data-transition* verfügbar[7]:

Wert	Funktion
slide	Die neue Seite fliegt von rechts ein und ersetzt somit die alte Seite. Dieser Wert ist als default eingestellt.
slideup	Die neue Seite fliegt von unten ein.
slidedown	Die neue Seite fliegt von oben ein.
pop	Zoomt die neue Seite ein.
fade	Die neue Seite wird mit tranzparenzen eingeblendet.
flip	Erstellt einen Flip-Effekt.
turn	Die Seite dreht sich rein.
flow	Die alte Seite zoomt out und die neue erscheint.
slidefade	Eine Kombination von slide und fade.
none	Es wird kein Effekt verwendet.

Tabelle 1: Mögliche Werte für Seitenübergänge

7 http://view.jquerymobile.com/1.3.1/dist/demos/widgets/transitions/

3.7 Komponenten (Widgets)

3.7.1 Dialogfenster

Bei einer Webapp dürfen natürlich Dialogfenster zur Interaktion nicht fehlen. Dafür bietet das Framework eigene Dialog-Komponenten. Mit diesen ist es möglich, Feedback zu geben oder auch Feedback zu erfragen.

Für ein Dialogfenster wird eine normale Page verwendet. Diese wird per Link aufgerufen und als Dialog dargestellt.

```
<div data-role="page" id="multipage-dialog">
            <div data-role="header"><h1>Multi-page dialog
window</h1></div>
            <div data-role="content">
                <p><a href="dialog.html" data-rel="back">OK</a></p>
            </div>
</div>
```

Um eine Page als Dialog aufzurufen, wird das Attribut *data-rel="dialog"* dem Link hinzugefügt.

```
<a href="#multipage-dialog" data-rel="dialog">Dialog öffnen</a>
```

3.7.2 Toolbars

Toolbars unterteilen die Anwendung optisch in unterschiedliche Bereiche z.B Header und Footer. Neben Titel oder Unternehmenslogo können Toolbars auch Navigationselemente enthalten.

3.7.2.1 Header- und Footer-Toolbars

Header- und Footer-Toolbars eignen sich für die Darstellung des Seitentitels, der Navigation oder anderen Bedienelementen wie Speicher-, Lösch-, oder Zurück-Buttons.

Diese bestehen in seiner einfachsten Form nur aus einer Überschrift.

```
<div data-role="header">
    <h1>Mein Titel</h1>
</div>
```

Das entscheidende Attribut hier ist *data-role="header"*. Dieses verwandelt den div-Container in eine Toolbar. Diese kann jetzt ganz einfach mit Buttons bestückt werden. Befindet sich ein Link in der Toolbar wird dieser automatisch als Button gerendert.

```
<a href="#" data-icon="check">Speichern</a>
```

Alternativ können Buttons auch gruppiert werden. Dabei erscheinen diese nebeneinander als zusammengehörendes Bedienelement.

```
<div data-role="controlgroup" data-type="horizontal">
        <a href="#" data-role="button" data-icon="delete">Löschen</a>
        <a href="#" data-role="button" data-icon="check">Speichern</a>
</div>
```

3.7.2.2 Navigation in der Toolbar

Um die Navigation der Anwendung im Header oder Footer zu platzieren wird eine Navbar verwendet. Diese wird mit dem Attribut *data-role* erstellt, wobei die einzelnen Navigationselmente in eine Liste gepackt werden.

```
<div data-role="header">
    <div data-role="navbar">
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="#">Firma</a></li>
            <li><a href="#">Kontakt</a></li>
        </ul>
    </div>
</div>
```

3.7.3 Layouts

Da wir für die Inhalte, durch die geringe Displaygröße, nur wenig Platz haben sollte die Anwendung diese übersichtlich präsentieren. Das Framework liefert uns dafür einige Komponenten, welche wir in diesem Kapitel betrachten wollen.

3.7.3.1 Grids

Mit Grid-Layouts können Inhalt nicht nur vertikal sondern auch horizontal angeordnet werden (ähnlich einer Tabelle mit Spalten und Zeilen). Erreicht wird dies über CSS-Klassen. JQM verwendet ein 5-spaltiges Grid-Layout welches über Buchstaben (a-d) angesprochen wird. Über die Klasse *ui-grid-[a-d]* wird angegeben wieviele Spalten das Grid haben soll. Dabei steht a für zwei Spalten bis zu d - fünf Spalten.

Die Zuordnung zu den einzelnen Spalten findet über die Klasse ui-block[a-d] statt. Hier deklariert a die erste Spalte.

```
<div class="ui-grid-a">
    <div class="ui-block-a"><strong>erste Spalte</strong></div>
    <div class="ui-block-b">zweite Spalte</div>
    <div class="ui-block-a"><strong>zweite Zeile, erste S.</strong></div>
    <div class="ui-block-b">zweite Zeile, zweite S.</div>
</div>
```

Um die Spalten entsprechend zu stylen gibt es bereits vorgefertigte Klassen welche verwendet werden können – dabei wird die Klasse „ui-bar" sowie „ui-bar-[a-z]" für das entsprechende Theme eingebunden.

```
<div class="ui-block-a"><div class="ui-bar ui-bar-c">Zelle</div></div>
```

3.7.3.2 Accordion

JQM ermöglicht es Inhalte zu erstellen, welche bei Berührung aufklappen. Dies ist besonders hilfreich, um den Inhalt übersichtlich in Teilbereiche zu untergliedern.

Dazu muss das Attribut *data-role="collapsible"* verwendet werden. Hier wird automatisch die Überschrift als klickbares Element definiert.

```
<div data-role="collapsible">
        <h3>Klick hier</h3>
        <p>Dann siehst du mich!</p>
    </div>
```

Um ein Accordion-Set zu erstellen (Darstellung als zusammenhängender Bereich), muss der entsprechende Bereich mit einem div-Block und *data-role="collapsible-set"* umschlossen werden.

3.7.4 Listen

Dies ist eine der praktischsten Komponenten. Listen werden vor allem zur Navigation verwendet. Das Framework bietet (un-)nummerierte Listen sowie Listen mit oder ohne Icons, Trennlinien oder Thumbnails.

Um eine normale jQM-Liste zu erstellen, reicht es das -Tag mit dem Attribut data-role="listview" auszuzeichnen.

```
<ul data-role="listview">
    <li><a href="#">Home</a></li>
    <li>Firma</li>
    <li>Referenzen</li>
    <li>Aktuelles</li>
    <li>Kontakt</li>
</ul>
```

Eine nummerierte Liste wird genauso erstellt, jedoch wird dazu anstatt dem HTML-Tag unordered-list *
* das ordered-list Tag verwendet *
...
*.

3.7.4.1 Individuelle Listen

Eine Liste kann mit auch mit anderen Inhalten bestückt werden. Einige besondere Listenarten werden hier aufgeführt.

Splitview-Listen

Hier können je Listeneintrag zwei Links hinterlegt werden. Mit dem zweiten Link wird rechts ein Bereich abgetrennt, der durch ein Icon verziert wird und separat klickbar ist.

```
<ul data-role="listview" data-split-icon="gear">
    <li>
        <a href="#item-detail">
            <h3>Liste</h3>
            <p>Mit einem zweiten Bereich.</p>
        </a>
        <a href="#purchase" data-rel="dialog">Bestellen</a>
    </li>
...
</ul>
```

Listen-Trenner

Mit folgendem Attribut können Listenelemente ganz einfach optisch getrennt werden:

```
<li data-role="list-divider">Devider</li>
```

Count-Bubbles

Ein wirklich nützliches Feature sind die sogenannten „bubbles". Diese ermöglichen es z.B. die Anzahl der Unterelemente einer Liste darzustellen.

```
<li>
        <a href="#outbox">Outbox
            <span class="ui-li-count">10</span>
        </a>
</li>
```

Thumbnails

Die nützlichen kleinen Vorschaubilder werden von JQM direkt in der passenden Größe angezeigt. Wird ein Bild eingebunden wird dies automatisch als Thumbnail dargestellt.

```
<li>
        <a href="#page">
            <img src="assets/img/bmwimage.png" >
            <h3>BMW</h3>
            <p>Automobile der besonderen Art.</p>
        </a>
</li>
```

Kann man dieses Bild auch als Icon verwenden? Kein Problem mit *class=„ul-li-icon*" in dem Bild-Tag verwandelt es sich automatisch in ein Icon.

3.7.5 Buttons

Buttons können genutzt werden, um ein Formular abzusenden oder auch um einen Link hervorzuheben. Diese Links verwandelt jQM in klickbare, userfreundliche Lösungen. Um einen Link in einen Button zu verwandeln, gibt es wie nicht anders zu erwarten war, das bereits bekannte Attribut *page-role* welches hier den Wert „*button*" hat.

```
<a href="#" data-role="button" data-inline="true">Button</a>
<a href="#" data-role="button">Fullsize-Button</a>
```

Absendeschaltflächen der Form *<input type="submit" .../>, <button ../>* werden automatisch als Buttons dargestellt. Soll dieser Mechanismus nicht zum tragen kommen, kann er mit *data-role="none"* für das entsprechende Tag deaktiviert werden.

Um Buttons anzupassen gibt es eigene Attribute. Diese ermöglichen z.B. die runden Ecken zu deaktivieren, ein Icon mit anzuzeigen, das Icon entsprechend zu positionieren oder Schatten und das Anzeigeverhalten zu ändern.

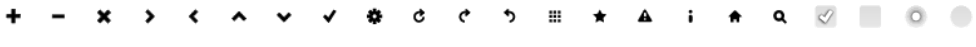
Attribute	Mögliche Belegung und Beschreibung
data-corners	true | false //Button mit runden Ecken
data-icon	 plus | minus | delete | arrow-r| arrow-l | arrow-u | arrow-d| check | gear | refresh | forward | back | grid | star | alert | info| home | search // Fügt dem Button ein Icon hinzu.
data-iconpos	left | right | top | bottom | notext //Legt die Position des Icons im Button fest. Default: left
data-iconshadow	true | false // Zeigt einen Schatten bei Icon an.
data-inline	true | false // Ob der Button nur um den Text gelegt werden soll.
data-shadow	true | false // Zeigt einen Schatten an.
data-theme	[a-z] mehr dazu unter dem Bereich Theme

Tabelle 2: Attribute für Buttons[8]

8 vgl. (Hadlock, K. ,2012, http://proquest.safaribooksonline.com/book/web-development/jquery/9780132947053/5dot-dialog-windows-and-buttons/ch05lev1sec2)

3.7.6 Formulare

Formulare sind ein zentraler Bestandteil um vom User Eingaben entgegenzunehmen. JQM unterstützt uns dabei, einheitliche Formulare zu generieren welche sich durch eine besondere Benutzerfreundlichkeit auszeichnen. So verwendet JQM auch viele der neuen HTML5-Typen, um z.B. eine korrekte Email-Adresse zu abufragen.

3.7.6.1 Input-Felder

Ein Input-Feld wird folgendermaßen definiert:

```
<form>
  <label for="number">Label zum Feld</label>
  <input type="number" name="number">
</form>
```

Neue HTML5 Inputtypen:[9]

number = Erwartet eine Nummer.

email = Erwartet eine Email-Adresse.

url = Eine Internetadresse wird erwartet.

tel = Eine Telefonnummer wird erwartet. Dazu wird der Ziffernblock eingeblendet.

time, *month*, *date* oder *datetime* = Erwartet entsprechend ein Datum / Zeit und öffnet ein entsprechendes Auswahlmenü.

3.7.6.2 Checkboxen und Radio-Buttons

Diese werden ebenfalls in dem üblichen HTML5-Markup angegeben. Das Framework generiert automatisch das passende look and feel.

```
<fieldset data-role="controlgroup">
  <label for="name">Name</label>
  <input type="checkbox" name="name" id="name" class="name" />
  <label for="my-alter">Alter</label>
  <input type="checkbox" name="my-alter" id="my-alter" class="my-
alter" />
</fieldset>
```

9 http://www.w3schools.com/html/html5_form_input_types.asp

Um mehrere Boxen als zusammengehörende Gruppe darzustellen, kann das Attribut *data-role="controlgroup"* im *<fieldset>*-Tag verwendet werden.

Eine weitere Option ist die horizontale Darstellung der Checkboxen:

```
<fieldset data-role="controlgroup" data-type="horizontal">
…
</fieldset>
```

3.7.6.3 Auswahlfelder (select-options)

Diese folgen dem Prinzip der bereits vorgestellten Elemente und basieren auf dem entsprechenden HTML-Tag.

```
<label for="select-prog">Buch wählen:</label>
<select name="select-cabin" id="select-prog">
   <option value="economy">Java</option>
   <option value="business">Patterns</option>
   <option value="first">Mobile World</option>
</select>
```

3.7.6.4 Slider

Slider sind eine Erweiterung der normalen Formularfelder. Diese eignen sich besonders zur Abfrage von Wertebereichen.

```
<label for="my-slider">Personen</label>
<input type="range" name="slider" id="my-slider" value="50" min="1"
max="10" />
```

3.7.6.5 Flip-Toogle Switches

Ein weiteres jQM spezifisches Tag welches es ermöglicht zwischen zwei Werten zu „switchen" z.B. für Ja/Nein-Abfragen.

```
<label for="flip-switch">Fahrradmitnahme?</label>
<select name="slider" id="flip-switch" data-role="slider">
    <option value="yes">Yes</option>
    <option value="no">No</option>
</select>
```

3.8 Themes und Swatches

Bei der Programmierung von jQM wurde auf eine leichte Anpassbarkeit des Designs geachtet.

Das Framework bringt bereits ein Theme mit, welches aus fünf verschiedenen Swatches besteht. Swatches sind Designvorlagen im Theme. Diese können für jedes Element oder global definiert werden. Hierfür gibt es das Attribut *data-theme*.

```
<div data-theme="c" data-role="header">
```

Durch Verwendung eines Buchstaben von a-e kann der entsprechende Swatch zugewiesen werden.

Mit der Webanwendung ThemeRoller können auch eigene Themes erstellt werden. Diese ist unter http://jquerymobile.com/themeroller/ zu finden.

Abbildung 4: Themes in jQuery Mobile (Hadlock, K. ,2012, http://proquest.safaribooksonline.com/book/web-development/jquery/9780132947053/11dot-theming-jquery-mobile/ch11lev1sec1)

3.9 Der Klick-Baukasten

Auf der Startseite von http://jquerymobile.com/ befindet sich außerdem noch ein sehr nützliches Werkzeug. Das Tool von Codiqa ermöglicht es eine jQuery Mobile Anwendung per grafischer Oberfläche zu erstellen. Somit müsste nicht eine Zeile Code selbst geschrieben werden. Interessant ist das Tool besonders, um sich schnell einen Prototypen zusammen zu klicken und dem Kunden so bereits einige Funktionalitäten zeigen zu können.

4 PrimeFaces

JavaServer Faces als Präsentationstechnologie zeichnet sich vor allem durch seine Komponenten aus. Es gibt neben den JSF-Standard Komponenten und Tags diverse andere Komponentenbibliotheken. Eine der populärsten in diesem Zusammenhang ist PrimeFaces. Mehr als 100 aufeinandere abgestimmten Komponenten vereinfachen das Erstellen von Applikationen enorm. Darüber hinaus bietet es auch eigene Ajax Erweiterungen gegenüber dem JSF-Standard sowie anpassbare Themes. Weitere Vorteile:[10]

- Besteht nur aus einem einzigen jar-File.
- Einfach zu nutzen.
- Keine Abhängigkeiten.
- Keine Konfiguration nötig.
- Nutzt zur Darstellung jQuery und HTML5.
- Unterstützt allen modernen Browser (IE6 ist kein moderner Browser!)

PrimeFaces ist sehr flexibel und unterstützt:

- CSS Overrides
- JS API
- Client Callbacks
- Ajax Callbacks

5 PrimeFaces Mobile

5.1 Was ist PrimeFaces Mobile?[11]

Mit PrimeFaces Mobile ist es möglich JSF-Seiten für mobile Geräte mit einem nativen Look&Feel auf Basis von jQM zu erstellen. Der UI-Baukasten zeichnet sich inbesondere aus durch:

- Ein mobiles RenderKit für JSF sowie PrimeFaces-Komponenten.
- Mobile JSF-Komponenten.
- Ein einheitliches backend model für Desktop- und Mobile-Web.

10 http://jsfatwork.irian.at/semistatic/primefaces.html

11 http://www.primefaces.org/documentation.html

- Unterstützung aller bekannten Plattformen.
- Ajax Features für eine native application experience.
- Keine Installation auf dem Client nötig.
- Die Power von jQuery Mobile.

5.2 Erste Schritte mit PrimeFaces Mobile

PrimeFaces Mobile besteht aus einem eigenen jar-File welches entweder manuell unter http://www.primefaces.org/downloads.html heruntergeladen oder per Maven in das eigene Projekt eingebunden werden kann. PrimeFaces Mobile benötigt außerdem mindestens die JAVA 5 runtime sowie JSF ab der Version 2.

Konfiguration für Maven in der pom.xml:

```
<dependency>
<groupId>org.primefaces</groupId>
<artifactId>primefaces-mobile</artifactId>
<version>0.9.4</version>
</dependency>

<repository>
<id>prime-repo</id>
<name>Prime Repo</name>
<url>http://repository.primefaces.org</url>
</repository>
```

Um PrimeFaces Mobile nutzen zu können muss zusätzlich die entsprechende PrimeFaces Mobile Komponenten-Bibliothek in die mobile Seite eingebunden werden.

```
xmlns:f="http://java.sun.com/jsf/core"
xmlns:p="http://primefaces.org/ui"
xmlns:pm="http://primefaces.org/mobile"
```

5.3 Das RenderKit

Wie bereits erwähnt bringt PrimeFaces Mobile ein eigenes RenderKit für JSF sowie PrimeFaces-Komponenten. Um das RenderKit von PrimeFaces Mobile zu aktivieren gibt es mehrere Möglichkeiten.

Die einfachste ist das setzen des f:view Tags mit der renderKitId.

```
<f:view renderKitId="PRIMEFACES_MOBILE">
//content
</f:view>
```

Damit wird ein neuer View generiert, welcher mit PrimeFaces Mobile gerendert wird.

Handelt es sich sowieso um eine rein mobile Applikation kann das RenderKit auch global für die gesamte Applikation in der faces-config.xml aktiviert werden.

```
<application>
<default-render-kit-id>PRIMEFACES_MOBILE</default-render-kit-id>
</application>
```

Optional kann man auch das RenderKit als Parameter in der URL übergeben. Dies ist vor allem zu Testzwecken gedacht.

Befinden sich in der Applikation mobile sowie non-mobile Seiten kann auch ein eigener Viewhandler geschrieben werden. Dazu wird einfach die calulateRenderKitID API überschrieben.

Das RenderKit bietet eigene Tags und rendert bestehende Komponenten um eine optimierte mobile Darstellung zu ermöglichen.

5.4 Pages

Eine Mobile-Seite ist ein XHTML basierter JSF View welcher aus einem oder mehreren PrimeFaces Mobile Views besteht. Ein View entspricht dabei einer mobilen Seite.

So kann das Prinzip von jQuery-Mobile mit mehreren Seiten in einem Dokument auch hier angewandt werden. Außerdem finden wir hier auch den Header und Footer.

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
```

```
  xmlns:p="http://primefaces.org/ui"
  xmlns:pm="http://primefaces.org/mobile">
  <f:view renderKitId="PRIMEFACES_MOBILE" contentType="text/html">

  <pm:page title="Mein Titel">
    <pm:view id="main">
    <pm:header title="Header" />
      <pm:content>
        Hier steht der Inhalt
    </pm:content>
    </pm:view>

  <pm:view id="page-two">
    <pm:header title="Titel Seite zwei" />
    <pm:content>
      Hier steht der Inhalt der zweiten Seite
    </pm:content>
   </pm:view>

  <pm:view id="page-three">
   <pm:header title="Titel Seite drei" />
    <pm:content>
      Hier steht der Inhalt der dritten Seite
    </pm:content>
  </pm:view>
</pm:page>
</f:view>
</html>
```

5.4.1 Navigation

Eine Navigation zwischen Views in einem Dokument:

Dabei wird kein neuer HTTP-Request abgesetzt, sondern der neue View einfach eingeblendet.

```
<h:outputLink value="#page-two">Seite 2</h:outputLink>
```

Dieses Konzept basiert auf dem JSF Navigations-Model:

```
<h:form>
     <p:commandButton value="Senden" action="#{bean.method}" />
     <p:commandButton value="Startseite" action="index" />
</h:form>
```

Deim drücken des „Senden"-Buttons wird auf den return Wert der hinterlegten Methode weitergeleitet:

```
public String method() {
  return "pm:pageTwo";
}
```

Wenn der Rückgabewert mit pm: beginnt, dann navigiert PrimeFaces Mobile zum View B nachdem der Ajax request fertig ist. Es ist auch möglich dies direkt anzugeben:

```
<p:commandButton value="Go" action="pm:pageTwo" />
```

Externe Ressourcen können direkt per URL verlinkt werden.

```
<h:outputLink value="http://www.google.de">Linktext</h:outputLink>
```

Navigieren über die Client-API:

```
PrimeFaces.navigate('#pageTwo', {
reverse: true|false //default ist es false
,transition: 'fade' //optionale transition
});
```

5.4.2 Seitenübergänge

Für die Seitenübergänge können alle Werte wie unter 3.6.2 Seitenübergänge beschrieben verwendet werden. Dabei werden diese hier der URL als Parameter angefügt.

```
<h:outputLink value="#viewTwo?transition=fade">Go</h:outputLink>
```

Dies funktioniert auch bei Aktionen über die hinterlegten Methoden:

```
public String method() {
```

```
    return “pm:pageTwo&transition=slide”;
}
```

5.4.3 Nutzung von Ajax

Eine mobile Seite lässt sich über das PrimeFaces Ajax Framework verbessern. Wichtig ist dabei nicht den ganzen View zu updaten sondern nur den Inhalt der Views. Deshalb sollte man, um nicht die gesamte Mobile UI zu verlieren, die in PrimeFaces mitgelieferte Ajax-Komponente verwenden.

```
<pm:page title="Mobile">
     <pm:view id="pageOne">
     <pm:header title="A" />
     <pm:content>
           <h:form>
                 <p:inputText value="#{bean.text}" />
                 <p:commandButton value="Update"
update=":formB:display" />
           </h:form>
     </pm:content>
     </pm:view>
     <pm:view id="pageTwo">
     <pm:header title="B" />
     <pm:content>
           <h:form id="formB">
                 <h:outputText id="display" value="#{bean.text}"/>
           <h:form>
     </pm:content>
     </pm:view>
</pm:page>
```

In diesem Beispiel soll der Inhalt der Komponente mit der id=“display“ beim Betätigen des Command-Buttons aktualisiert werden. Da JSF beim Suchen der Komponente von einem relativem Pfad ausgeht, erwartet er diese Komponente unter den Kindern seines NamingContainers. Einer dieser NamingContainer ist das Form. Um nun ein Element daraus zu referenzieren, muss dessen vollqualifizierender Name angegeben werden.

5.5 Mobile Komponenten

5.5.1 Toolbars

5.5.1.1 Header- und Footer-Toolbars

Pendant zu jQuery-Mobile: 3.7.2.1 Header- und Footer-Toolbars

Diese befinden sich am Anfang bzw. Ende der Seite.

```
<pm:view id="main">
<pm:header>
  Dies ist mein Header
</pm:header>
//...
</pm:view>
```

5.5.1.2 Navbar

Pendant zu jQuery-Mobile: 3.7.2.2 Navigation in der Toolbar

Diese können wieder im Header oder im Content platziert werden.

```
<pm:view id="navbar">
<pm:header title="NavBar">
  <pm:navBar>
    <p:button value="Home" icon="home" href="#main?reverse=true"
styleClass="ui-btn-active"/>
    <p:button value="Info" icon="info" href="#main?reverse=true" />
    <p:button value="Suche" icon="search" href="#main?reverse=true" />
  </pm:navBar>
</pm:header>
…
```

Die styleClass setzt mit *ui-btn-active* den aktuellen Navigationspunkt aktiv.

5.5.2 Switch

Dies sind die unter jQuery-Mobile beschriebenen Flip-Toogle Switches (3.7.6.5).

```
<pm:switch value="#{bean.value}" onLabel="Ja" offLabel="Nein"/>
```

5.5.3 Slider (InputRange)

Neben der Unterstützung des HTML Inputtyps *range* bietet PrimeFaces Mobile noch einen eigenen range-Tag an. Dieser zeigt den Slider von jQuery Mobile (3.7.6.4 Slider) an.

```
<pm:inputRange value="#{bean.value}" />
```

5.6 Unterstüzung der Standard-JSF Bibliothek

5.6.1 PanelGrid

Das aus JSF verfügbare panelGrid wird in einer mobil angepassten Form ohne Tabellen dargestellt. Der Wert von *columns* kann nur von 1 bis 5 liegen. Dies ist in dem Grid von jQM mit seinen fünf Spalten begründet (dort waren als Werte a-e möglich vgl. 3.7.3.1 Grids).

```
<h:panelGrid columns="4">
  <h:outputText value="Neuer" />
  <h:outputText value="Lahm" />
  <h:outputText value="Dante" />
  <h:outputText value="Alaba" />
  <h:outputText value="Götze" />
  <h:outputText value="Schweinsteiger" />
  <h:outputText value="Robben" />
  <h:outputText value="Ribery" />
</h:panelGrid>
```

5.7 Unterstützung von PrimeFaces

5.7.1 Buttons

Buttons aus den PrimeFaces Komponenten werden für mobile Endgeräte optimiert und um das PrimeFaces Mobile Navigationsmodel erweitert. Diese können somit wie gewohnt verwendet werden (3.7.5 Buttons).

```
<p:button value="Home" icon="home" href="#main" />
```

Command-Buttons

Diese werden für mobile Endgeräte optimiert und um das PrimeFaces Mobile Navigationsmodel erweitert.

```
<p:commandButton value="With Icon" icon="check" update="othercomponent"/>
```

5.7.2 Buttons gruppieren

Damit können Buttons gruppiert werden. Sollen die Buttons anstatt untereinander horizontal dargestellt werden, kann das Attribut *orientation="horizontal"* in den Tag *pm:buttonGroup* eingefügt werden.

```
<pm:buttonGroup>
     <p:commandButton value="Ja" />
     <p:commandButton value="Nein" />
</pm:buttonGroup>
```

5.7.3 Listen

Listen eignen sich besonders um Informationen übersichtlich darzustellen und als Navigationskonzept. Pendant zu jQuery-Mobile: 3.7.4 Listen.

Bei DataLists im mobilen Kontext müssen jedoch folgende Aspekte beachtet werden:

- Ajax Pagenierung wird nicht unterstützt,
- um die DataList zu aktualisieren, müssen die PrimeFaces Ajax-Komponenten verwendet werden (JSF Ajax ist nicht möglich),
- umschließe den Inhalt einer Spalten-Komponente falls diese markierbar sein soll (spezifiziert in UIData)

Normale Liste:

```
<p:dataList>
  <h:outputText value="Java" />
  <h:outputText value="C#" />
```

```
  <h:outputText value="C++" />
</p:dataList>
```

Liste zur Navigation:

```
<p:dataList>
  <h:outputLink value="#home">Home</h:outputLink>
  <h:outputLink value="#company">Firma</h:outputLink>
  <h:outputLink value="#contact">Kontakt</h:outputLink>
</p:dataList>
```

Eigene Listen

Listen können mit weiteren Inhalten wie „bubbles" oder Thumbnails angereichert werden.

```
<p:dataList value="#{ringBean.players}" var="player">
<p:graphicImage value="/images/barca/#{player.photo}" />
<h3><h:outputLink value="#main">#{player.name}</h:outputLink></h3>
<p>#{player.position}</p>
<h:outputText styleClass="ui-li-count" value="#{player.number}" />
</p:dataList>
```

Listen durchsuchen

Listen können außerdem gefiltert werden, durch das Setzen von f:attribute.

```
<f:attribute name="filter" value="true" />
```

Listen trennen

Als separator kann *p:separator* verwendet werden, um Listenelemente zu trennen.

5.7.4 Formulare

5.7.4.1 Input-Felder

Mit *p:inputText* kann man ein normales Input-Feld erstellen. Durch die Erweiterung mit dem Attribut *type="value"* können unterschiedle Eingabewerte für die Felder definiert werden. Dies richtet sich nach den HTML5 spezifischen.

```
<p:inputText id="txt" type="text" value="#{bean.value}" />
```

Zusätzlich kann mit *p:inputTextarea* eine Textarea zur Eingabe von mehrzeiligem Text ausgegeben werden. Pendant zu jQuery-Mobile: 3.7.6.1 Input-Felder.

5.7.4.2 Checkboxen und Radio-Buttons

Bei den Checkboxen gibt es eine Boolean-Checkbox sowie die normalen Checkboxen mit mehreren Auswahlmöglichkeiten.

Eine Boolean-Checkbox kann folgendermaßen erstellt werden:

```
<p:selectBooleanCheckbox value="#{true}" itemLabel="Check me" />
```

Eine Checkbox mit mehreren Auswahlmöglichkeiten kann über *p:selectManyCheckbox* definiert werden. Analog dazu ist die vorgehensweise bei Radio-Buttons über *p:selectOneRadio. Pendant zu jQuery-Mobile: 3.7.6.2 Checkboxen und Radio-Buttons.*

```
<p:selectCheckboxMenu value="#{bean.selectedOptions}" label="Movies">
  <f:selectItems value="#{bean.options}" />
</p:selectCheckboxMenu>
```

5.8 Theming

Auch hier ist es möglich das Design anzupassen. Von den fünf Swatches welche von jQM bereits zur Verfügung gestellt werden, können unterschiedliche zugewiesen werden(3.8 Themes und Swatches).

```
<pm:view id="main" swatch="a">
    <pm:header title="Header" swatch="b" />
      <pm:content>
```

Um ein eigens gestaltetes Theme von ThemeRoller einzubinden geht mal folgendermaßen vor:

1. Die Dateien unter WebRoot entpacken.
2. Das Standard-Theme deaktivieren in der web.xml

```
<context-param>
  <param-name>primefaces.mobile.THEME</param-name>
  <param-value>none</param-value>
</context-param>
```

3. Mit dem facet pre-init das neue Theme (CSS-File) einbinden

```
<pm:page>
  <f:facet name="preinit">
    <link type="text/css" rel="stylesheet"
href="#{request.contextPath}/themes/hm.min.css" />
  </f:facet>
</pm:page>
```

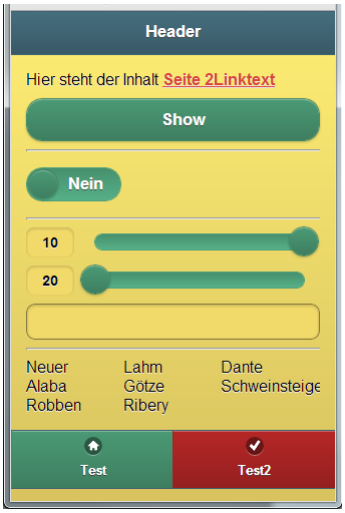


Abbildung 5: Eigenes Theme

6 Fazit

Wie bereits in der Einleitung erwähnt, werden mittlerweile die mobilen Geräte zur Internetnutzung bevorzugt. Dies zeigt sich auch dadurch, dass sogar Zuhause dem handlichen Mobilgerät der Vorzug gegeben wird. Für die IT ist dies natürlich von Vorteil, da Anwendungen für mobile Geräte entwickelt werden müssen.

Mit PrimeFaces Mobile wird das Erstellen von mobilen Seiten dabei einfach und schön zugleich. Hier wurden zwei starke Technologien vereint. Die Stärke von jQuery Mobile als plattformunabhängige Präsentationstechnologie sowie PrimeFaces Mobile mit allen Vorteilen von PrimeFaces und JSF.

Jedoch täuscht dies nicht darüber hinweg, dass man auch bestimmte Kompromisse eingehen muss. Verwendet man jQuery Mobile zeigt sich, dass das Framework keine Fehler verzeiht. Ein Attribut an einem falschen Element angebracht und das Layout zerschießt sich. Wenn man das Layout an eigene Bedürfnisse anpassen will (z.B. Symbole neben Eingabefeldern positionieren), welche nicht durch jQM abgedeckt werden, wird es schon schwieriger. Hier muss man sich dann auch mit HTML, CSS und Javascript auskennen und sich in den dynamisch erstellten Markup einarbeiten. Nichtsdestotrotz geht das Arbeiten mit dem Framework leicht von der Hand und man hat schnell eine mobile Benutzeroberfläche welche auf allen Plattformen läuft und durch seine Effekte beeindruckt.

Bei der Arbeit mit PrimeFaces zeigt sich, dass man nicht umherkommt sich erst einmal intensiv mit der JSF-Komponenten-Bibliothek sowie Prime-Faces zu beschäftigen. War diese Hürde erst einmal geschafft hat man auch mit Prime-Faces Mobile schnelle Ergebnisse. Der Versuch eine zuvor erstellte jQuery-Mobile Anwendung mit Prime-Faces Mobile nachzubauen kann dennoch Probleme machen. Das gute ist, das Prime-Faces Mobile auf bestehende Komponenten setzt und diese dann mit dem eigenen Renderer darstellt. Die wichtigsten Komponenten wurden auf Basis von jQuery Mobile dann als erweiterte mobile Komponenten umgesetzt.

Abschließend lässt sich mit den Worten von Optimus Prime aus dem Kinofilm „Transformers", *„Am Ende dieses Tages, wird einer stehen, einer wird fallen."*[12] sagen, dass das Projekt zumindest im Sinne der User Experience gute Chancen hat am Ende nicht zu fallen.

12 Optimus Prime gab Prime-Faces seinen Namen, da der Entwickler unter diesem Pseudonym aktiv ist. http://www.filmzitate.info/suche/film-zitate.php?film_id=2489

7 Abbildungsverzeichnis

8 Tabellenverzeichnis

9 Literaturverzeichnis

Mobile Web:

http://de.statista.com/*

Mobile user interface design:

http://en.wikipedia.org/wiki/User_interface_desige*

http://mobile.smashingmagazine.com/2012/07/12/elements-mobile-user-experience/*

http://mobile.smashingmagazine.com/2011/05/02/a-user-centered-approach-to-mobile-design/*

jQuery-Mobile:

http://jquerymobile.com/ *

http://www.w3schools.com/html/html5_form_input_types.asp*

Hadlock, K. (2012). jQuery mobile - develop and design, Peachpit Press

Dutson, P. (2013). Sams teach yourself jQuery mobile in 24 hours,Sams Pub.

Firtman, Maximiliano R. (2012), jQuery mobile - up and running, O'Reilly Media

PrimeFaces /Mobile:

http://www.primefaces.org/*

http://jsfatwork.irian.at/semistatic/primefaces.html *

http://www.filmzitate.info/suche/film-zitate.php?film_id=2489*

**Alle Webseiten aufgerufen zwischen dem 24.04.2012 und 09.05.2012.*