



Hochschule für Angewandte  
Wissenschaften Hamburg

*Hamburg University of Applied Sciences*

**Department Medientechnik**

---

**Konzeption und Erstellung einer  
Windows 8 App mit  
Webtechnologien bei ZEIT ONLINE  
zum Thema; Nachrichten visuell  
erleben**

---

von Malte Modrow

Matrikel-Nr. 1975941

**Bachelorthesis (B.Sc.)**

Studiengang Media Systems

Erstprüfer: Prof. Dr. Andreas Plaß

Zweitprüfer: Arne Seemann (M.Sc.)

Hamburg, 3. September 2013

## **Zusammenfassung**

deutsch

## **Abstract**

englisch

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
<b>2. Windows 8</b>	<b>2</b>
2.1. Neues Bedienkonzept . . . . .	2
2.2. Die Windows 8-Philosophie . . . . .	4
2.3. Windows RT . . . . .	5
2.4. Das Ökosystem . . . . .	5
2.4.1. Windows Phone . . . . .	6
2.4.2. Microsoft App Stores . . . . .	6
<b>3. Konzeption der App</b>	<b>8</b>
3.1. ZEIT ONLINE . . . . .	8
3.1.1. Die Webseiten . . . . .	8
3.1.2. Die iPad App . . . . .	10
3.1.3. Die Webapp . . . . .	10
3.1.4. Die iPhone App . . . . .	11
3.2. Ziele der App - Was wird dargestellt . . . . .	12
3.3. Graphical User Interface . . . . .	13
3.3.1. Flaches Navigationssystem . . . . .	13
3.3.2. Hierarchisches Navigationssystem . . . . .	14
3.3.3. Menü und Features . . . . .	16
3.4. Umsetzungsmöglichkeiten in Visual Studio . . . . .	16
3.4.1. Native App . . . . .	17
3.4.2. Windows 8 App mit Webtechnologien . . . . .	17
<b>4. Umsetzung der App</b>	<b>19</b>
4.1. Rastervorlage . . . . .	19
4.1.1. HTML Dateien . . . . .	20
4.1.2. Javascript Dateien . . . . .	20
4.1.3. CSS Dateien . . . . .	21
4.2. Datenanbindung . . . . .	21
4.2.1. ZEIT ONLINE Datenbestand . . . . .	21
4.2.2. Daten verfügbar machen . . . . .	22
4.3. Darstellung der Daten . . . . .	25
4.3.1. Die Hubansicht . . . . .	25
4.3.2. Die Ressortansicht . . . . .	27
4.3.3. Die Artikelansicht . . . . .	27
4.4. Features . . . . .	29
4.4.1. Artikeltitel ausblenden . . . . .	30
4.4.2. Schriftgröße verändern . . . . .	30

4.4.3. Semantic Zoom . . . . .	31
4.5. Entwicklungschronologie . . . . .	33
<b>5. Evaluierung</b>	<b>35</b>
5.1. Fokusgruppe . . . . .	35
5.2. Ablauf . . . . .	35
5.3. Ergebnisse . . . . .	37
5.3.1. Intuitivität der App . . . . .	37
5.3.2. Der Nur-Bilder-Modus . . . . .	38
<b>6. Fazit</b>	<b>40</b>
<b>Literatur</b>	<b>41</b>
<b>A. Detaillierte Evaluierung Bild-Inhalt Relation</b>	<b>i</b>
<b>B. CD Inhalt</b>	<b>iii</b>

# Abbildungsverzeichnis

1.	Der Startbildschirm von Windows 8 . . . . .	2
2.	Die Geste zum Aufrufen der Charms-Menüleiste. . . . .	3
3.	Die Gesten zum Durchwechseln der Apps. . . . .	4
4.	Die Geste zum Öffnen der unteren und oberen Menüleiste. . . . .	4
5.	Ein möglicher Startscreen von Windows Phone 8. . . . .	6
6.	Die Verfügbarkeit der Top 100 iOS Apps . . . . .	7
7.	Die Homepage von ZEIT ONLINE. . . . .	9
8.	Die mobile Webseite von ZEIT ONLINE. . . . .	9
9.	Die iPad App . . . . .	10
10.	Die Webapp . . . . .	11
11.	Die iPhone App . . . . .	11
12.	Das flache Navigationssystem . . . . .	13
13.	Das flache Navigationssystem . . . . .	14
14.	Das Schema des „Semantic Zoom“ . . . . .	15
15.	Die Single-Page-Navigation . . . . .	19
16.	Die Projektstruktur in Visual Studio. . . . .	20
17.	Die Hubansicht. . . . .	25
18.	Die Einzelressortübersicht . . . . .	27
19.	Die Artikelansicht. . . . .	28
20.	Die untere Menüleiste mit der Funktion zum Artikeltitel ausschalten. . . . .	30
21.	Das Reisen-Ressort ohne Artikeltitel. . . . .	31
22.	Die Schriftgröße-Buttons . . . . .	31
23.	Die Semantic Zoom Ansicht . . . . .	32
24.	Die Testbilder aus dem Politik- und Wissen-Ressort. . . . .	36

# Tabellenverzeichnis

1.	Übersicht der Testpersonen. . . . .	35
----	-------------------------------------	----

## Listingsverzeichnis

1.	Das XML eines Teaserelements . . . . .	22
2.	Initialisierung der Binding-List. . . . .	23
3.	Auszug aus dem Ressorts Array. . . . .	23
4.	Parsen und Speichern der Daten. . . . .	24
5.	Die wichtigsten Markupelemente der Hubansicht. . . . .	26
6.	Template bei der Listview registrieren. . . . .	26
7.	Setzen der Datenquelle und des Layouts. . . . .	27
8.	Das Einfügen von Titel sowie TeaserText und Bild. . . . .	28
9.	Vermeidung von Infoboxen und Twitternachrichten sowie Hyperlinks in den Artikeln. . . . .	29
10.	Der Semantic Zoom Wrapper. . . . .	33

## Abkürzungsverzeichnis

<b>XAML</b>	Extensible Application Markup Language . . . . .	17
<b>CMS</b>	Content Management System . . . . .	21
<b>XML</b>	Extensible Markup Language . . . . .	21
<b>XSLT</b>	Extensible Stylesheet Language Transformations . . . . .	21
<b>HTML</b>	Hypertext Markup Language . . . . .	17
<b>XHR</b>	XMLHttpRequest . . . . .	23
<b>WinJS</b>	Windows Library for JavaScript . . . . .	18
<b>GUI</b>	Graphical User Interface . . . . .	13

## 1. Einleitung

Das Konsumieren von Nachrichten erfolgt zunehmend über mobile Endgeräte. In der Klasse der kleinen Endgeräte, zum Beispiel Smartphones, sind das herstellerübergreifende Betriebssystem Android von Google und iOS von Apple führend, das Betriebssystem Windows Phone von Microsoft ist bisher weniger stark verbreitet.

Bei den größeren mobilen Endgeräten, den modernen Tablet-Computern, waren zunächst nur die Betriebssysteme Android und iOS vertreten. Erst seit Oktober 2012 versucht Microsoft mit Windows 8 hier nachzuziehen, indem es ein Betriebssystem entwickelt hat, dass die Systembasis für herkömmliche Programme, aber zugleich auch für touchoptimierte Anwendungen bietet. Vor diesem Hintergrund gibt es für Windows 8 sehr viel weniger Apps als für die beiden anderen genannten Betriebssysteme, die deutlich früher an den Start gegangen sind.

Das online Angebot der Wochenzeitung DIE ZEIT ist ZEIT ONLINE. Für die Inhalte der ZEIT ONLINE Webseite gibt es bisher neben der mobilen Webseite *mobil.zeit.de* nur eine App für mobile Geräte, und zwar für das iPhone. Mit der Entwicklung der in dieser Arbeit beschriebenen App für das Betriebssystem Windows 8 sollen die Inhalte von ZEIT ONLINE für weitere Endgeräte erschlossen werden. Des Weiteren wird untersucht, inwieweit das Konsumieren von Nachrichten über eine reine visuelle Bildbetrachtung für den Leser möglich und sinnvoll ist.

Als Grundlage wird zunächst auf ausgewählte Aspekte von Windows 8 eingegangen, auf die Windows 8-Philosophie und das neue Bedienkonzept. Anschließend wird die aktuelle Situation bezüglich der mobilen Apps bei ZEIT ONLINE betrachtet. Im weiteren Verlauf der Arbeit wird detailliert auf die Konzeption der App und deren Umsetzung eingegangen: Was soll dargestellt werden? Wie soll es dargestellt werden? Und wie wird es entwickelt?

Abschließend wird im Rahmen der Evaluierung die intuitive Handhabung der App untersucht sowie auf Aspekte des rein visuellen Erlebens von Nachrichten eingegangen.

## 2. Windows 8

Seit dem 26. Oktober 2012 ist Microsofts aktuellstes Betriebssystem Windows 8 für die breite Öffentlichkeit verfügbar. Dieses unterscheidet sich grundlegend von seinem Vorgänger Windows 7. Microsoft verfolgt mit Windows 8 den Ansatz, das gleiche Betriebssystem sowohl für Desktop- als auch für Tablet-Computer<sup>1</sup> zu verwenden. Es besitzt nach wie vor den bekannten Desktop von Windows 7. Äußerlich sind hier nur kleine Änderungen vorgenommen worden. So besitzt z.B. der Windows Explorer ein neues, kontextsensitives Ribbon-Menü<sup>2</sup>. Neu ist hingegen das "Modern User Interface"(Modern UI, früher Metro), eine für Touch-Gesten optimierte Oberfläche. Diese kann jedoch ebenso auf herkömmliche Weise mit Maus und Tastatur bedient werden. Es können, im Gegensatz zum Desktop, keine herkömmlichen Programme im Modern UI ausgeführt werden. In der Modern UI Oberfläche laufen nur Programme (Apps), die speziell für Windows 8 entwickelt wurden und über den Microsoft Store (siehe Sektion ?? auf Seite ??) erhältlich sind. Es soll nun ein etwas detaillierterer Blick auf die neue Oberfläche und dessen Eigenheiten geworfen werden.

### 2.1. Neues Bedienkonzept

Beim Hochfahren von Windows 8, erwartet den User ein Interface, das er so vom Vorgänger Windows 7 nicht kennt, nämlich viele bunte, viereckige Kacheln in verschiedenen Größen. Auf dem Startbildschirm von Windows 8 sind die verschiedenen Apps analog Abbildung 1 angeordnet.



Abbildung 1: Der Startbildschirm von Windows 8.

Dies können Desktop-Programme oder Modern UI Apps (Windows-Store) sein, welche nur über den Microsoft Store zu installieren sind. Der bekannte Desktop-Modus ist in diesem Sinne auch „nur“ ein spezielles Programm (App), welches in der neuen Oberfläche gestartet und ausgeführt werden kann.

<sup>1</sup>Tablet-Computer: kleiner, flacher, tragbarer Computer mit einem Touchscreen. Es wird im Weiteren der Einfachheit halber die englische Version „Tablet“ benutzt.

<sup>2</sup>Auch bekannt als Menüband oder Multifunktionsleiste.

## 2. Windows 8

Es gibt einige grundlegende Touch- und Mausgesten, um in der Modern UI Oberfläche zwischen den Apps zu navigieren oder um Optionen oder Einstellungen aufzurufen. Wenn der User sich auf einem Tablet befindet und über den rechten Bildschirmrand von rechts nach links wischt (swipen), öffnet sich am rechten Bildschirmrand die von Microsoft „Charms“ genannte Menüleiste (siehe Abbildung 2). Hier befinden sich die Schaltflächen: Suchen, Teilen, Start , Geräte und Einstellungen. Um die Charms-Bar auf einem Gerät ohne Touch-Unterstützung aufzurufen, muss die Maus in die obere oder untere rechte Ecke des Bildschirms geführt werden.



Abbildung 2: Die Geste zum Aufrufen der Charms-Menüleiste.

Es können zwei verschiedene Gesten benutzt werden, um bequem zwischen den geöffneten Apps zu wechseln. Abbildung 3 zeigt die beiden Möglichkeiten. Um eine Liste, wie sie die Abbildung auf der linken Seite darstellt, angezeigt zu bekommen, wird von links nach rechts über den linken Bildschirmrand gewischt. Anschließend wird ohne den Finger hochzunehmen zurück in Richtung des Bildschirmrandes gewischt. Bei der Bedienung mit der Maus wird die Maus zunächst in die obere oder untere linke Ecke bewegt, um anschließend nach unten bzw. nach oben zur Bildschirmmitte geführt zu werden.

Die zweite Möglichkeit besteht darin, die Apps direkt in der Reihenfolge, wie sie zuletzt aktiv waren, wieder in den Vordergrund zu bringen. Dazu wird von links nach rechts über den linken Bildschirmrand gewischt. Der rechte Teil von Abbildung 3 zeigt diese Geste.

Die letzte grundlegende Geste wird benutzt, um z.B. Optionen, Einstellungen oder eine Navigationsleiste aufzurufen. Hierzu wird über den oberen oder den unteren Bildschirmrand in Richtung Bildschirmmitte gewischt. Wird diese Geste ausgeführt, öffnen sich die untere und die obere Menüleiste, unabhängig davon, ob der User die Geste am oberen oder am unteren Bildrand ausführt. Wenn es in der App oben und unten Menüleisten gibt, öffnen sich immer beide. Es ist nicht möglich, z.B. nur die obere Leiste zu öffnen. In der unteren Menüleiste befinden sich typischer Weise Einstellungen für die aktuelle Ansicht, in welcher sich die App gerade befindet. Sofern es eine Leiste am oberen Bildrand gibt, ist in dieser häufig eine Art von Navigation

## 2. Windows 8

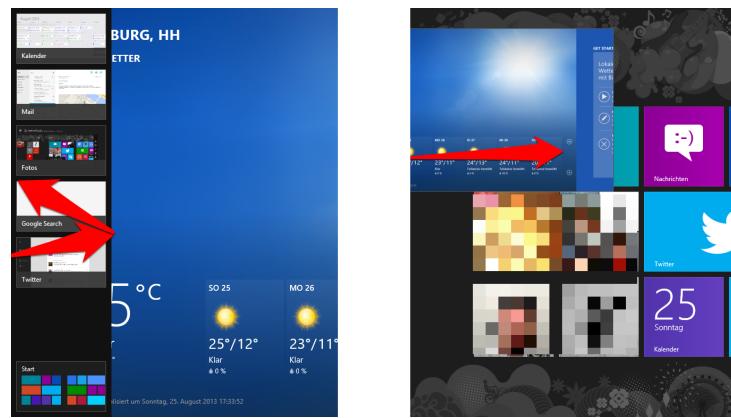


Abbildung 3: Die Gesten zum Durchwechseln der Apps.

untergebracht. Abbildung 4 zeigt die obere und untere Menüleiste am Beispiel der vorinstallierten Wetter App.



Abbildung 4: Die Geste zum Öffnen der unteren und oberen Menüleiste.

## 2.2. Die Windows 8-Philosophie

Windows 8 ist wohl der radikalste Umbruch bei den Microsoft-Betriebssystemen seit Windows 95. Es sollen hier einige Aspekte beleuchtet werden, warum Microsoft diesen Weg eingeschlagen hat.

Viele Features, die heute mit einem Windows PC verbunden werden wie z.B. der Desktop, die Taskleiste oder das Startmenü, wurden ursprünglich in Windows 95 eingeführt. Damals hatte Windows noch keinen Browser. Mit der Zeit gewann das Internet immer mehr an Bedeutung, und der Browser rückte mehr und mehr in den Mittelpunkt der vernetzten Computernutzung. Das Betriebssystem hingegen veränderte sich kaum. Mit Windows 8 versucht Microsoft diesen Schritt nachzuholen und das Betriebssystem um den vernetzten User herum zu konzipieren. Eine der Hauptaufgaben übernehmen hierbei die sogenannte *Live Tiles* auf dem Startbildschirm. Sie zeigen stets aktuelle Informationen, ohne dass der User den Browser öffnen muss (Pachal, 2012).

## 2. Windows 8

*There's tons of stuff on the Internet and your PC basically has this little straw — Internet Explorer — to see all this. We didn't think that should be the case. The whole PC should be about that. Part of what the Start screen is really about is making all this activity — these people that you care about, and all this information — sort of explode so you're immersed in it. —Sam Moreau<sup>3</sup>*

Sam Moreau bringt es gut auf den Punkt. Es gibt so viel zu entdecken im Internet, doch bisher hatte der User nur einen kleinen „Strohhalm“ , den Browser, um dies alles zu entdecken. Diese Funktion soll in Windows 8 der gesamte PC übernehmen. Alle Informationen, die dem User wichtig sind, sollten möglichst einfach einsehbar und erreichbar sein.

### 2.3. Windows RT

Windows RT ist ein separates Windows Betriebssystem, welches für PCs oder Tablets mit ARM-Architektur optimiert ist. Die größten Vorteile dieser Geräte sind normalerweise ein leichteres Gewicht und eine längere Akkulaufzeit gegenüber den Geräten, die mit einer x86/x64 Architektur ausgestattet sind. Auf den ersten Blick sieht Windows RT, von der Bedienoberfläche her, genau so aus wie Windows 8. Der größte und signifikanteste Unterschied zu Windows 8 ist jedoch der, dass aufgrund der ARM-Architektur auf Windows RT nur Programme und Apps aus dem Windows-Store installiert und ausgeführt werden können. Das Installieren von herkömmlichen Programmen, wie es Windows 8 zulässt, ist nicht möglich. Windows RT verfügt zwar über einen Desktop, wie er auch in Windows 8 vorhanden ist, allerdings laufen hier nur die vorinstallierten Office Produkte von Microsoft sowie der Microsoft Browser *Internet Explorer*. Außerdem können nur speziell für Windows RT zertifizierte Geräte benutzt werden, das gilt u.a. auch für Mäuse und Tastaturen. (Microsoft, 2013c)

### 2.4. Das Ökosystem

Kaufte man früher einen PC, ein Notebook oder ein Handy, ging es bei der Abwägung der Kaufkriterien meist ausschließlich um Hersteller und Hardwarespezifikationen. In einer mehr und mehr mobilen und *mehrgerätigen* Welt, wie wir sie heute erleben, kommt zu der Kaufentscheidung noch ein weiterer entscheidender Punkt hinzu. Welches Ökosystem passt am besten zu mir? Denn um auf mehreren Geräten wie z.B. Notebook, Tablet und Smartphone stets seine Daten aktuell und synchron zu halten, muss vorher genau bedacht werden, ob das gewählte Ökosystem den eigenen Anforderungen entspricht. Fragen, die zum Ökosystem gestellt werden können, sind z.B.: Wie sieht es mit der Anzahl und Verfügbarkeit von Apps im jeweiligen Store aus? Gibt es eine Musikstreaming-Lösung? Wie sieht es mit Filmen und Büchern aus? Wie

---

<sup>3</sup>Director of design and research for Windows

## 2. Windows 8

sehen die Dienste zum Speichern von Daten in der Cloud aus? Es fällt auf, dass heute nicht mehr nur ein Gerät gekauft wird, sondern dass zu diesem Gerät oder diesen Geräten meist ein ganzes Ökosystem mit geliefert wird, unabhängig davon, ob man sich für Apple, Google, Microsoft oder Amazon entscheidet.

Mit der Einführung von Windows 8 hat Microsoft neuen Schwung in sein gesamtes Ökosystem gebracht. Es wurden verschiedene Dienste vereinheitlicht und zusammengefasst. Das Microsoft Ökosystem umfasst heute u.a. Windows 8, Windows RT, das Smartphone Betriebssystem Windows Phone 8, den Cloudspeicherdiens Skydrive, den Streamingdienst xBox Music und den E-Mail Dienst outlook.com. Viele dieser Dienste existierten schon vor der Einführung von Windows 8. Aber erst mit dessen Einführung wurden die Dienste und Betriebssysteme einer einheitlichen „Designkur“ unterzogen und enger aufeinander abgestimmt. Außerdem wird im Gegensatz zu vorher nur noch ein Benutzerkonto benötigt, um diese Dienste nutzen zu können.

### 2.4.1. Windows Phone

Windows Phone ist das Smartphone Betriebssystem von Microsoft. Die aktuelle Version ist Windows Phone 8. Es besitzt genau wie Windows 8 eine Kachel-Oberfläche. Obwohl Windows Phone 8 und Windows 8 auf dem gleichen Kernel basieren, können auf dem Smartphone keine Windows 8 Apps ausgeführt werden. Umgekehrt laufen Windows 8 Apps nicht unter Windows Phone. Auch beim Erstellen der Apps kann nicht gleichzeitig für beide Plattformen entwickelt werden. Abbildung 5 zeigt einen Startbildschirm von Windows Phone 8.



Abbildung 5: Ein möglicher Startscreen von Windows Phone 8.

### 2.4.2. Microsoft App Stores

Eines der wichtigsten Elemente im Ökosystem eines Unternehmens ist der Store. Hier werden Apps für die Geräte mit dem jeweiligen Betriebssystem angeboten. Bei Apple

## 2. Windows 8

ist es der *App Store*, bei Google der *Google Play Store* und bei Microsoft ist es zum einen der *Windows Store* für Windows 8- und Windows RT Apps und zum anderen der *Windows Phone Store* für Smartphones mit Windows Phone. Oft fällt eine Entscheidung gegen ein bestimmtes Gerät aus der Windows-Welt mit der Begründung, es gäbe zu wenig Apps im Windows Store.

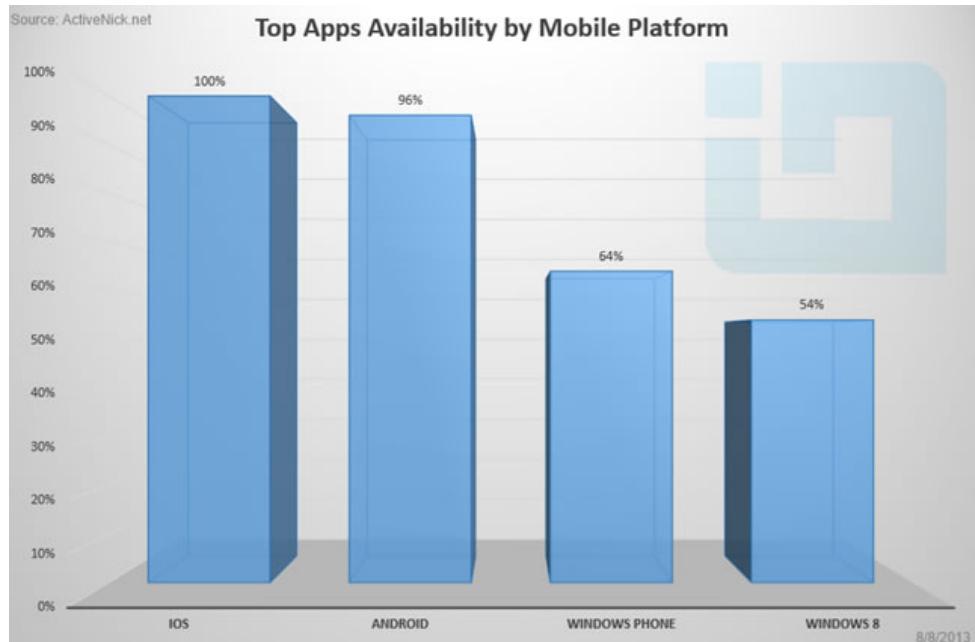


Abbildung 6: Verfügbarkeit der Top 100 iOS Apps auf anderen Systemen (WinBeta, 2013).

Abbildung 6 zeigt die Verfügbarkeit der Top 100 iOS Apps<sup>4</sup> auf den anderen großen Systemen wie Android, Windows Phone und Windows 8. Hiernach sind im Windows Phone Store 64% der Top 100 Apps vorhanden und im Windows 8 Store 54%. Gezählt wurden hier nur offizielle Apps und keine Apps von Drittanbietern. In absoluten Zahlen hat iOS ca. 900.000 Apps im Appstore, Google über 1 Million Apps im Play Store, Windows Phone ca. 160.000 Apps und Windows 8 ca. 110.000 Apps im Windows Store (WinBeta, 2013).

Microsoft bemüht sich, die fehlenden Apps wie z.B. *Instagram*, *Pinterest*, *Instapaper* oder *Readability* schnellstmöglich nachzuliefern. Dabei wird den Entwicklern teilweise sogar Geld geboten, damit sie eine offizielle „große“ App entwickeln (WinBeta, 2013).

<sup>4</sup>Liste: <http://www.infragistics.com/community/blogs/nick-landry/archive/2013/08/06/top-100-apps-availability-on-ios-android-windows-phone-and-windows-8.aspx>

## 3. Konzeption der App

In diesem Abschnitt wird zunächst ein Blick auf die ZEIT ONLINE Webseite und auf bereits vorhandene ZEIT ONLINE Apps geworfen. Anschließend geht es darum, die Konzeption für eine Windows 8 Nachrichten App darzustellen und zu erläutern. Dabei gilt es darzulegen, warum und mit welchem Hintergrund Entscheidungen zu Gunsten der einen oder der anderen technischen oder konzeptionellen Methode ausfallen. Dazu müssen zunächst die Ziele der App dargelegt werden. Anschließend muss festgelegt werden, mit welcher Technologie bzw. Programmiersprache gearbeitet werden soll. Zuletzt wird ein detaillierterer Blick auf das Herzstück der App geworfen, das Navigationskonzept. Hierbei ist eine der wichtigsten Fragen: „Wie kann ich auch bei ggf. großen Datenmengen eine übersichtliche, intuitive Struktur schaffen, die sich in das Gesamtkonzept von Windows 8 eingliedert?“.

Damit die zu erstellende App im Folgenden einen griffigeren Namen hat, wird an dieser Stelle die Namensfindung erläutert, sie heißt „ZEIGHT“ (gesprochen: Zeit). Der Name setzt sich zusammen aus „ZEIT“ und der englischen Zahl „eight“. Hintergrund ist, dass eine „Zeit“-App für Windows 8 (eight) entwickelt wird.

### 3.1. ZEIT ONLINE

ZEIT ONLINE ist das Internetangebot der Wochenzeitung DIE ZEIT. Berichtet wird über tagesaktuelle Ereignisse von einer eigenen Redaktion, die größtenteils in Berlin sitzt. Den Themenschwerpunkt bildet das Politik-Ressort. Um die ZEIT ONLINE Inhalte sowie auch die Inhalte der gedruckten ZEIT möglichst vielen Nutzern zugänglich zu machen, betreibt ZEIT ONLINE verschiedene Webseiten und Apps. Im folgenden soll ein Überblick über die bereits vorhandenen Webseiten und Apps gegeben werden.

#### 3.1.1. Die Webseiten

Das Hauptaugenmerk bei ZEIT ONLINE liegt auf der herkömmlichen Webseite. Hier werden aktuelle Nachrichten dargestellt, die laufend aktualisiert werden.

Abbildung 7 zeigt den obersten Teil der Homepage. Im oberen Bereich der Abbildung ist die Navigationsleiste zu erkennen. Sie zeigt die verschiedenen Haupt-Ressorts von ZEIT ONLINE. Klickt der User auf ein Ressort in der Navigation, landet er auf der jeweiligen sogenannten *Centerpage* des Ressorts. Diese sieht im oberen Teil genauso aus wie die Homepage. Es gibt an erster Stelle ein großes Aufmacherbild oder Video und darunter meist mehrere kleinere *Teaserelemente*. In der rechten Spalte sind die zurzeit meistgelesenen Artikel sowie die Artikel mit den meisten Nutzerkommentaren aufgelistet. Auf der Startseite gilt diese Liste über alle Ressorts hinweg, auf den jeweiligen Centerpages der Ressorts ist die Auflistung auf das jeweilige Ressort beschränkt.

### 3. Konzeption der App

The screenshot shows the ZEIT ONLINE homepage. At the top, there's a navigation bar with links like 'Abo', 'Shop', 'ZEITmagazin', 'ZEITCampus', 'ZEITGeschichte', and 'ZEITWissen'. Below the navigation is the ZEIT ONLINE logo. A search bar with 'ZEIT ONLINE durchsuchen' and a 'Suchen' button is present. There are also links for 'Partnersuche', 'Immobilien', 'Automarkt', 'Jobs', and 'Reiseangebote'. A banner at the top says 'START POLITIK WIRTSCHAFT MEINUNG GESELLSCHAFT KULTUR WISSEN DIGITAL STUDIUM KARRIERE LEBENSART REISEN MOBILITÄT SPORT'. Below this, a message says 'Zuletzt aktualisiert: vor 8 Minuten | Aktuelle Themen: Ägypten | Drogen | Syrien'. On the right, there are buttons for 'Anmelden' and 'Registrieren'. The main content area features a large photo of Christian Wulff with his hands clasped. Below the photo, the text 'PROZESS Christian Wulff muss vor Gericht' is displayed. A brief summary follows: 'Das Landgericht Hannover hat den früheren Bundespräsidenten die Eröffnung des Verfahrens mitgeteilt. Ab 1. November soll sich Wulff wegen Vorteilsnahme verantworten.' Below this, there are three small images with captions: 'WULFF-ANKLAGE Der Rechtstaat funktioniert', 'CHRISTIAN WULFF Rückzug ausgeschlossen', and 'EX-BUNDESPRÄSIDENT Rächt sich ein Parteidreß an Wulff?'. To the right, there's a sidebar titled 'MEISTGELESEN' with a list of five articles and 'MEISTKOMMENTIERT' with another list.

Abbildung 7: Die Homepage von ZEIT ONLINE.

Es gibt auf der ZEIT ONLINE Webseite nicht nur Nachrichtenbeiträge zu lesen. Daneben werden außerdem u.a. Bilderstecken, Videos, Spiele und Quizze angeboten.

Da zunehmend immer mehr Leser die Webseite vom Smartphone aus aufrufen, gibt es unter *mobil.zeit.de* eine für Smartphones optimierte mobile Webseite. In der Startansicht der mobilen Webseite werden alle Teaser-elemente der Desktop Seite dargestellt. Außerdem werden darunter 3 Elemente aus jedem Ressort aufgeführt. Der Benutzer kann auch auf der mobilen Webseite direkt in ein Ressort wechseln und bekommt dort alle Elemente des jeweiligen Ressorts angezeigt. Auf Abbildung 8 sind von links nach rechts, die Startseite, die Artikelansicht und das Navigationsmenü für die Ressorts dargestellt.

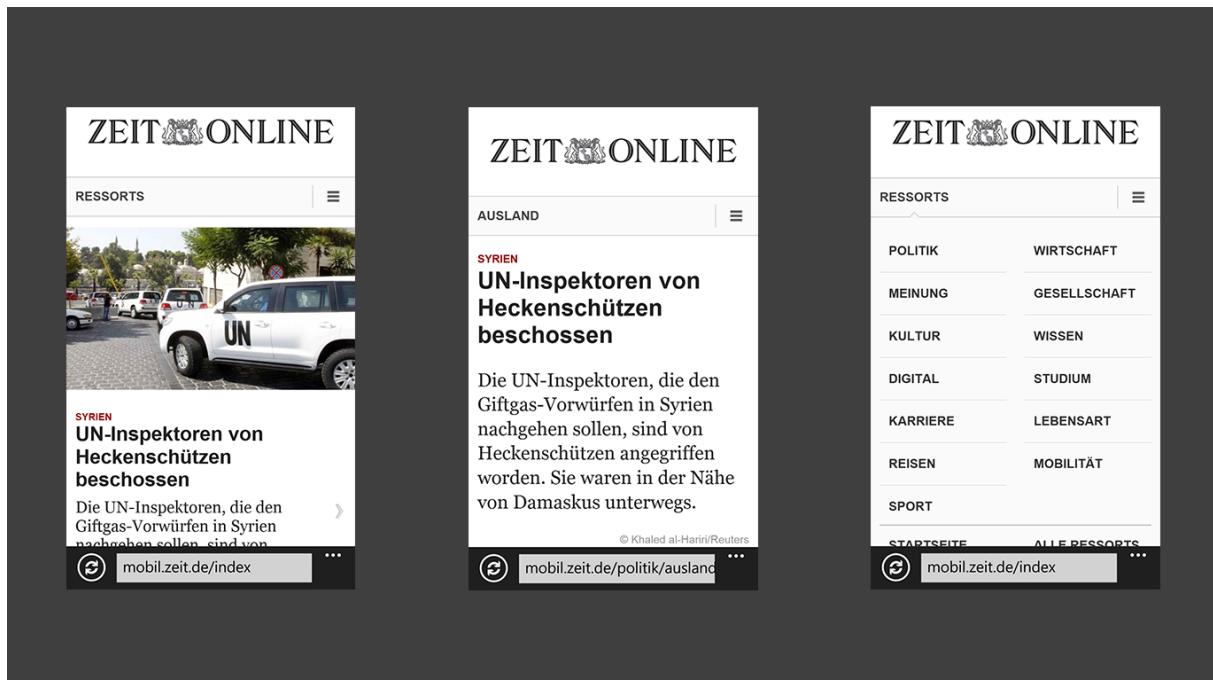


Abbildung 8: Die mobile Webseite von ZEIT ONLINE.

### 3. Konzeption der App

#### 3.1.2. Die iPad App

Neben den ZEIT ONLINE Webseiten existieren mehrere Apps für verschiedene Geräte. Dazu gehört auch die App für das iPad von Apple. In der iPad App können keine Inhalte von ZEIT ONLINE angesehen werden. Stattdessen bietet sie jede Woche die digitale Ausgabe der gedruckten ZEIT und das ZEIT Magazin an. Die Ausgaben können sowohl im Abonnement als auch einzeln erworben werden. Eine eigene iPad Redaktion bereitet jede Woche die Artikel der gedruckten Ausgabe der ZEIT für die digitale Ansicht auf. Es werden z.B. Fotostrecken oder Audiokommentare mit eingebunden.

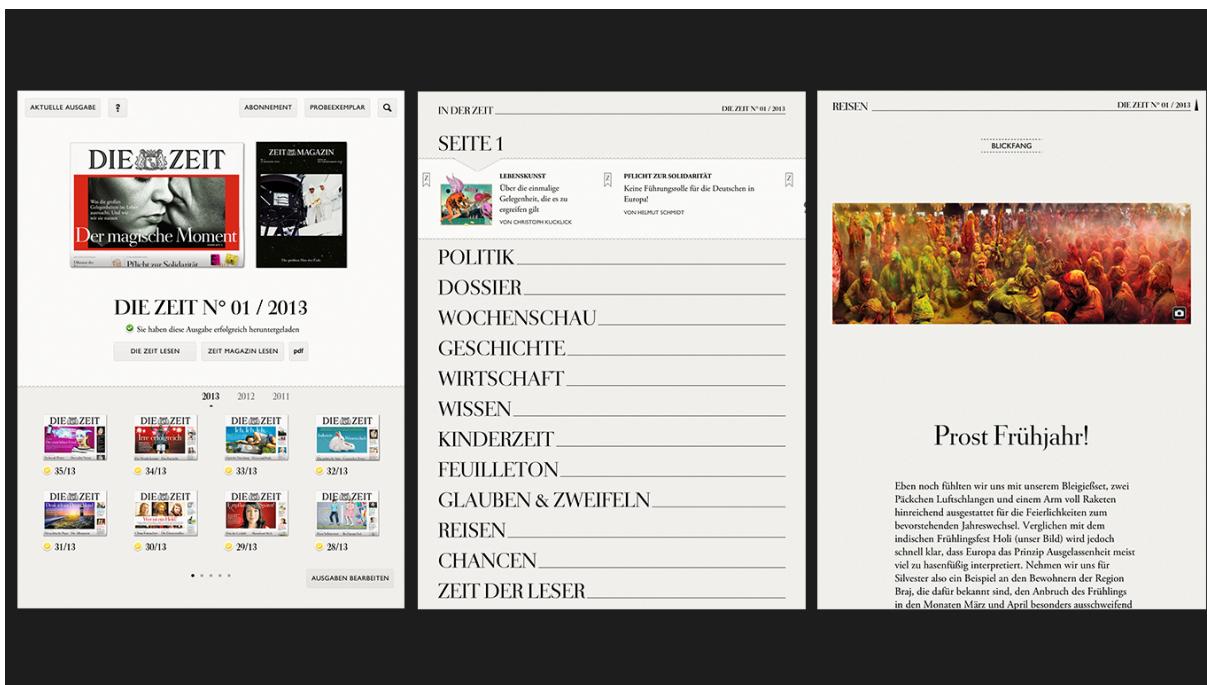


Abbildung 9: Die iPad App zeigt die aufbereiteten Artikel der gedruckten Ausgabe der ZEIT.

Abbildung 9 zeigt links die Ausgabenverwaltung der App. Hier können die Ausgaben erworben und heruntergeladen werden. In der Mitte ist das Inhaltsverzeichnis dargestellt. Hier ist gut zu sehen, in wieweit sich die Ressorts von ZEIT ONLINE und DIE ZEIT unterscheiden. Rechts auf der Abbildung ist ein zufälliger Artikel zu sehen. Findet der User einen Artikel interessant oder möchte ihn zum späteren Lesen speichern, kann er ihn im Bereich *Meine Zeit* ablegen.

#### 3.1.3. Die Webapp

Eine weitere App ist die Webapp. Sie ist im Grunde eine abgespeckte Version der iPad App. Es können ebenfalls keine ZEIT ONLINE Inhalte konsumiert werden, sondern nur die digitalen Ausgaben der ZEIT erworben werden. Die Webapp ist im Browser unter [webapp.zeit.de](http://webapp.zeit.de) erreichbar. Des Weiteren gibt es einen Android- und einen Kindle-

### 3. Konzeption der App

Wrapper, sodass die App auch über den *Google Play Store* sowie den *Amazon Store* zu beziehen ist.

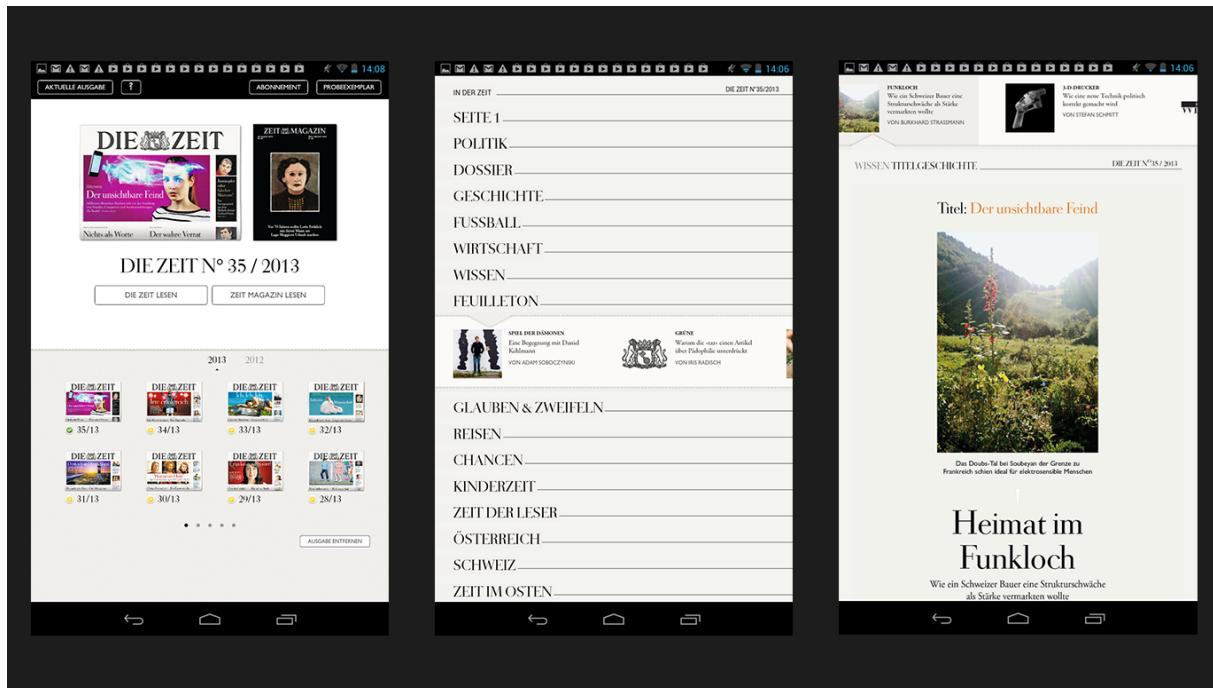


Abbildung 10: Die Webapp zeigt genau wie die iPad App die aufbereiteten Artikel der gedruckten ZEIT.

Abbildung 10 zeigt die Webapp auf einem Android Tablet (Nexus 7). Auf den ersten Blick scheint sie identisch mit der iPad App. In der Webapp können jedoch keine Lesezeichen, wie es in der iPad App möglich ist, gesetzt werden. Außerdem kann hier aus einem Artikel heraus nicht direkt in ein Ressort gesprungen werden. Auch dies ist bei der iPad App möglich.

#### 3.1.4. Die iPhone App

In der iPhone App werden dem User sowohl Inhalte von ZEIT ONLINE als auch die digitale PDF-Ausgabe der gedruckten Zeit angeboten.

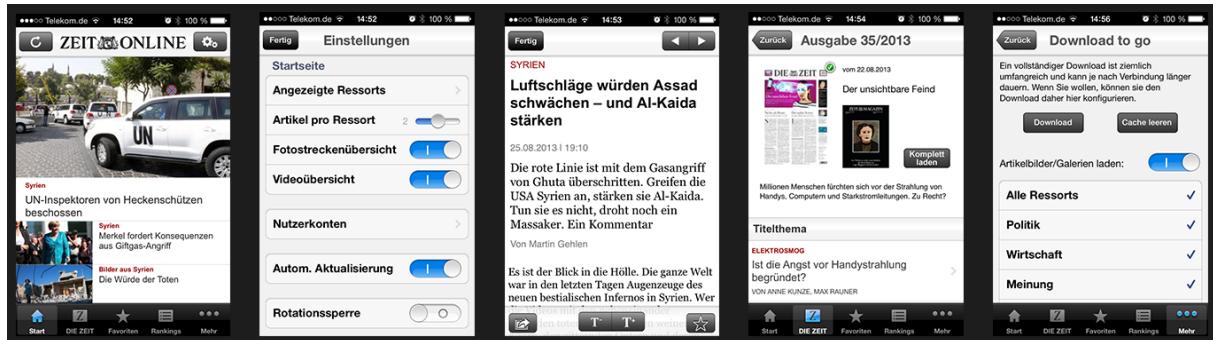


Abbildung 11: Die iPhone App zeigt sowohl Inhalte von ZEIT ONLINE als auch von der ZEIT.

### *3. Konzeption der App*

Die inhaltliche Struktur ähnelt der der mobilen Webseite. Allerdings kann der User die iPhone App nach seinen Vorlieben anpassen. So kann z.B. eingestellt werden, wie viele Artikel pro Ressort auf der Startseite angezeigt werden sollen.

Abbildung 11 zeigt eine Bildstrecke der iPhone App. Das zweite Bild von links zeigt einige der möglichen Einstellungen, die der User vornehmen kann. Das Bild ganz rechts zeigt ein weiteres Feature der App. Es können ausgewählte Artikel heruntergeladen werden und so für den Offline-Betrieb gespeichert werden.

## **3.2. Ziele der App - Was wird dargestellt**

Das übergeordnete Ziel der App ist es, eine Nachrichten Applikation für Windows 8 zu erstellen, die den Inhalt der ZEIT ONLINE Webseite auf für den Nutzer ansprechende und intuitive Art und Weise darstellt. Das Layout und die Darstellung der Inhalte sollen sich an das sogenannte „Look and Feel“ von Windows 8 anpassen und sich daran orientieren.

Der Fokus bei den Inhalten liegt auf den Artikeln selbst und den jeweiligen Aufmacher bzw. Teaser Bildern. Andere Inhalte wie z.B. Bildergalerien, Infografiken, Blogartikel oder Quizze werden von der App nicht erfasst und nicht dargestellt, da es in der Fragestellung schwerpunktmäßig um die Relation zwischen den Aufmacherbildern und den dahinter liegenden Artikeltexten geht. Die App erhebt in dieser Hinsicht keinen Anspruch darauf, die gesamten redaktionellen Inhalte von ZEIT ONLINE darzustellen, sondern versteht sich eher als explorative Applikation im Sinne der Fragestellung.

Gleichzeitig soll eine Umgebung geschaffen werden, die es erlaubt Untersuchungen anzustellen, inwieweit es möglich ist, allein durch das Betrachten der Aufmacherbilder auf den Inhalt der jeweiligen Artikel zu schließen (siehe Sektion 5.3.2 auf Seite 38). Der User soll die Möglichkeit haben, die standardmäßig vorhandenen Artikeltitel auszublenden, um so, wenn gewünscht, einen rein visuellen Eindruck der Artikelbilder zu bekommen.

Außerdem soll die App dazu dienen zu erforschen, wie eine Nachrichten App in der Modern UI Oberfläche von Windows 8 erstellt wird und welche design- und auch funktionstechnischen Vorgaben von Microsoft vorhanden sind, das heißt wie eine App mit Nachrichteninhalten aus Microsofts Sichtweise auszusehen hat.

Zusammengefasst soll eine Windows 8 App erstellt werden, welche die Artikel der ZEIT ONLINE Webseite darstellt. Dieses übergeordnete Ziel umfasst drei Teilziele.

1. Das User Interface der App ist ansprechend gestaltet
  - Look & Feel entspricht Windows 8
  - Die Nutzer können die App intuitiv bedienen
2. Die Umsetzung eines „Nur-Bilder-Modus“ ist vorhanden

### 3. Konzeption der App

3. Für zukünftige Windows 8 Entwicklungen bei ZEIT ONLINE ist Know How über die App-Entwicklung mit Webtechnologien vorhanden

## 3.3. Graphical User Interface

Damit der User ein für ihn angenehmes und flüssiges Nutzungserlebnis hat, ist es notwendig, sich über das Graphical User Interface (GUI) Gedanken zu machen, gerade wenn es sich um eine App handelt, in der es viele verschiedene Inhaltsbereiche gibt. Der User soll sich möglichst intuitiv durch die Inhalte bewegen können. Außerdem muss dem User auf der Einstiegsebene ein guter Überblick über die vorhandenen Inhalte gegeben werden, damit er von dort aus zielgerichtet weiter navigieren kann, ohne lange suchen zu müssen. Hierzu ist muss ein passendes Navigationskonzept gefunden werden. Microsoft nennt in seinen Richtlinien für Windows Store Apps (Modern UI Apps) grundsätzlich zwei Möglichkeiten, wie die Navigation umgesetzt werden kann. Diese werden nachfolgend beschrieben.

### 3.3.1. Flaches Navigationssystem

Das flache Navigationssystem wird in vielen Windows Store Apps verwendet, häufig in Spielen, Browzern oder in Apps zum Erstellen von Dokumenten. Es zeichnet sich dadurch aus, dass sich die Inhalte auf der gleichen hierachischen Ebene befinden. Dieses System eignet sich dann, wenn ein schneller Wechsel zwischen wenigen Seiten oder Registerkarten möglich sein soll (Microsoft, 2013a).

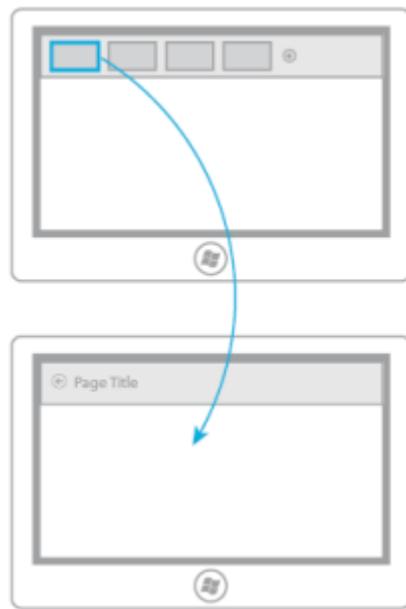


Abbildung 12: Das flache Navigationssystem (Microsoft, 2013a).

Abbildung 12 zeigt, wie das flache Navigationssystem funktioniert. Am oberen Rand befindet sich eine nicht immer sichtbare Navigationsleiste mit den verschie-

### 3. Konzeption der App

denen Registerkarten oder Seiten. Die Leiste wird angezeigt, wenn der User vom unteren oder oberen Bildrand wischt (siehe Sektion 2.1). Wenn der User die Seite wechseln möchte, geschieht dies direkt über die Navigationsleiste, d.h. es gibt meist keinen permanenten Zurück-Button.

Ein flaches Navigationssystem eignet sich nicht für eine Nachrichten App wie die, die in dieser Arbeit konzipiert und umgesetzt werden soll, weil es zu viele Bereiche (Ressorts) gibt, die sich strukturell zudem sehr ähneln. Hier ist es angebracht, ein hierarchisches Navigationssystem zu verwenden.

#### 3.3.2. Hierarchisches Navigationssystem

Die meisten Windows Store-Apps benutzen ein hierarchisches Navigationssystem, Microsoft nennt es Hubnavigationsmuster. Es ist geeignet, um große Inhaltssammlungen zu ordnen und sie für User benutzerfreundlich aufzubereiten. Der Schlüssel dazu ist die Unterteilung des Inhalts in verschiedene Detailebenen (Microsoft, 2013a).

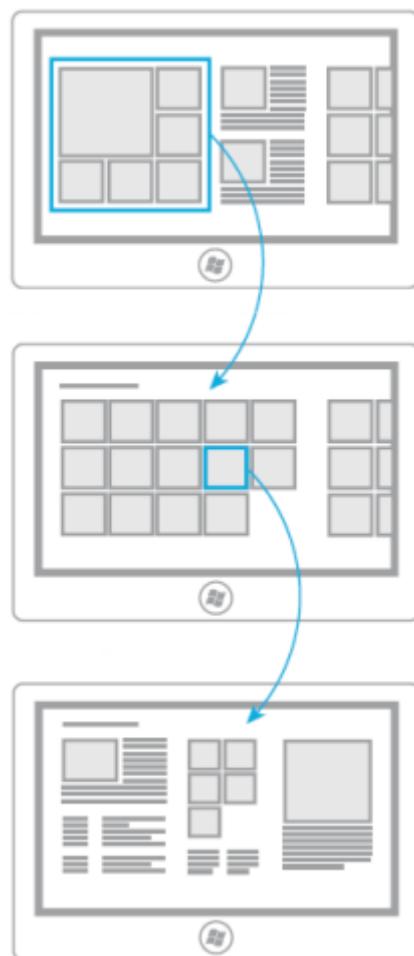


Abbildung 13: Das hierarchische Navigationssystem (Microsoft, 2013a).

Das Schema eines hierarchischen Systems ist in Abbildung 13 dargestellt. Der Einstiegspunkt in die App ist die sogenannte „Hubansicht“, hier ganz oben in der Abbil-

### 3. Konzeption der App

dung zu sehen. In der Hubansicht wird aus den vielen großen Bereichen der App jeweils ein kleiner Teil gesammelt und dargestellt, sodass sich der User vorstellen kann, was ihn im jeweiligen Bereich erwartet. Die Anordnung, in welcher der Auszug der Inhalte dargestellt ist, muss nicht für alle Bereiche gleich sein, es können verschiedene Templates für den Seitenaufbau hinterlegt werden. Es kann das Nutzungserlebnis verbessern und vielfältiger gestalten, wenn unterschiedliche Darstellungen z.B. in Höhe, Breite oder Anzahl der Objekte für die Bereiche genutzt werden. Die Hubansicht wird horizontal gescrollt, d.h. weiterer Inhalt befindet sich hinter dem rechten Rand und kann entweder durch das Scrollrad der Maus oder auf einem Tablet durch das Swipen nach links in das Sichtfeld des Users gebracht werden.

Die mittlere Darstellung in Abbildung 13 zeigt die zweite Detailstufe des hierarchischen Systems. Hier werden alle Elemente eines Bereichs dargestellt.

Im unteren Bereich der Abbildung ist schließlich die letzte Detailstufe zu sehen. Es handelt sich hierbei um den eigentlichen Inhalt, z.B. einen Artikel. Es ist ebenfalls möglich, von der Hubansicht direkt in die Detailansicht eines Elements zu wechseln.

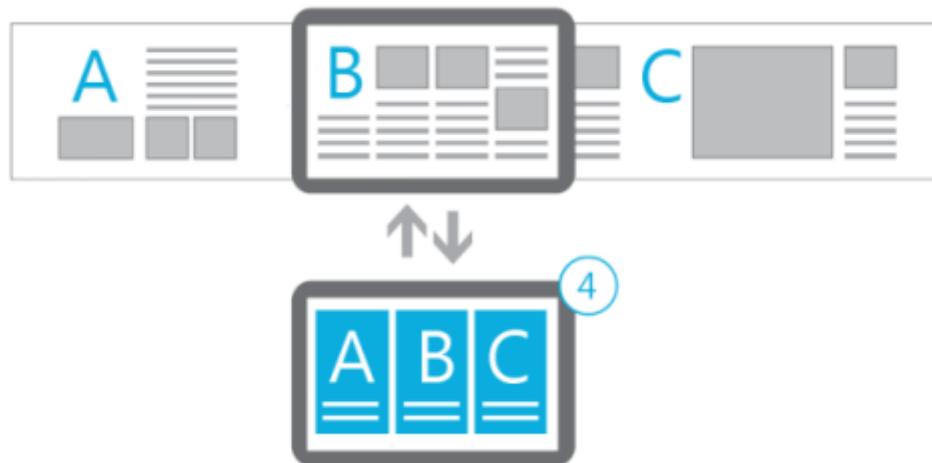


Abbildung 14: Das Schema des „Semantic Zoom“ (Microsoft, 2013a).

Befinden sich in der Hubansicht, trotz der Zusammenfassung der Bereiche noch zu viele Bereiche und es dauert zu lange, bis zum Ende der Darstellung zu scrollen, kann noch eine weitere Ebene „davor“ gelegt werden. Das Konzept nennt Microsoft „Semantic Zoom“ und ist in Abbildung 14 schematisch dargestellt. Oben in der Abbildung ist die Hubansicht dargestellt, unten die zusätzliche „Semantic Zoom“ Ebene. In der Praxis kann der User auf diese Weise die Inhalte noch weiter vereinfachen und zusammenfassen lassen. Im Fall dieser App soll es einen semantischen Zoom geben, um ein flüssigeres und schnelleres Navigieren z.B. ganz zum Ende der Hubansicht zu ermöglichen, da es wahrscheinlich über zehn verschiedene Ressorts geben wird. Um am Desktop-PC mit Maus diese Ansicht aufzurufen, muss der User auf ein kleines Minuszeichen am unteren rechten Rand klicken. Am Tablet wird der semantische Zoom

### 3. Konzeption der App

mit der „Pinch to Zoom (out)<sup>5</sup>“ Geste aufgerufen. Klickt der User auf ein Element in dieser zusätzlichen Navigationsebene, wird er direkt zum jeweiligen Ausschnitt in die Hubansicht navigiert. Die „Semantic Zoom“ Ansicht kann von jedem Punkt aus der Hubansicht aufgerufen werden.

#### 3.3.3. Menü und Features

Da die App einen explorativen Charakter hat, werden die Funktionen und Features auf das Nötigste beschränkt. Es soll eine untere App-Leiste geben, in der in Abhängigkeit von der gerade gewählten Ansicht verschiedene Optionen angezeigt werden. Diese Leiste ist nicht dauerhaft zu sehen und kann durch einen Rechtsklick aufgerufen werden, falls mit der Maus gearbeitet wird. Auf einem Tablet geschieht dies durch Swipen vom oberen oder unteren Bildschirmrand.

In der Startansicht sowie in der Ressortübersicht soll es am unteren Bildrand die Möglichkeit geben, die Titel aller Elemente aus- bzw. auch wieder einzublenden, um so dem User alternativ ein rein visuelles Erleben der Artikelbilder zu ermöglichen. In der Artikelansicht soll der User die Möglichkeit haben, die Schriftgröße des Artikels zu vergrößern, zu verkleinern oder wieder auf ihren Startwert zu setzen, da die App eventuell auf Monitoren mit verschiedenen großen Auflösungen ausgeführt wird oder die Sehkraft des Users nicht mehr ausreicht, um eine normal große Schrift zu erkennen und zu lesen. So wird ein gewisses Maß an Barrierefreiheit gewährleistet.

## 3.4. Umsetzungsmöglichkeiten in Visual Studio

Vor dem Entwickeln einer Windows 8 App muss entschieden werden, mit welcher Technologie bzw. mit welcher Programmiersprache entwickelt werden soll. Die Rede ist hier von einer App, die in der Modern UI Oberfläche von Windows 8 läuft und für diese konzipiert ist. Es handelt sich nicht um eine klassische .NET oder WIN32 Anwendung für Windows. Um eine Modern UI App zu entwickeln, müssen zwei Dinge zwingend vorhanden sein. Zum einen wird Windows 8 selbst und zum anderen die neueste Version von Visual Studio, Visual Studio 11<sup>6</sup>, benötigt. Visual Studio steht in der Express Version für Windows 8 kostenlos zur Verfügung. Des Weiteren muss zwischen der nativen Umsetzung und der Implementierung mit Webtechnologien entschieden werden. Es soll zunächst erläutert werden, was die beiden Begriffe bedeuten und in welcher Weise und welchem Zusammenhang sie bei der Entwicklung einer Windows 8 Modern UI App üblicherweise gebraucht werden.

---

<sup>5</sup>Zwei Finger (Berührungspunkte), die sich auf dem Bildschirm aufeinander zu bewegen.

<sup>6</sup>Visual Studio 2011 war die aktuellste Version beim Erstellen dieser Arbeit.

#### 3.4.1. Native App

Der heutige Begriff „native App“ unterscheidet sich in einigen Punkten von der früheren oder ursprünglichen Verwendung des Begriffs. Früher sprach man von einer nativen App, wenn direkt auf die Ressourcen der Maschine zugegriffen wurde, wie z.B. Maschinencode, der direkt von der CPU ausgeführt wird. Heute wird eine App oftmals schon als nativ deklariert, wenn es sich nicht um eine Webapp handelt. Eine sinnvolle Definition liegt zwischen diesen beiden Varianten. Eine App ist dann nativ, wenn sie geräte-, betriebssystem- oder laufzeitumgebungsabhängig ist. Sie ist für ein spezielles Gerät entwickelt und kann nur auf diesem ausgeführt werden. Dabei kann sie alle geräte- oder betriebssystemspezifischen Funktionen nutzen, es ist jedoch nicht relevant wie nah an der Hardware tatsächlich programmiert wurde (O'Brian, 2013).

In Visual Studio können native Windows 8 Apps u.a. mit den Programmiersprachen C++, C# oder Visual Basic erstellt werden. Für das Design bzw. das Aussehen der App wird die Markupsprache - Extensible Application Markup Language (XAML) - verwendet. Es gibt zwei Möglichkeiten, wie das XAML erstellt werden kann. Es kann entweder von Hand geschrieben oder automatisch generiert werden. Zum automatischen Generieren lassen sich per Drag & Drop Elemente wie z.B. Buttons und andere Schaltflächen aus einer Werkzeugpalette direkt auf die „App-Leinwand“ ziehen und dort verschieben oder nach Belieben anordnen.

#### 3.4.2. Windows 8 App mit Webtechnologien

In Visual Studio können nicht nur Apps mit den klassischen Programmiersprachen, wie in der vorigen Sektion beschrieben, erstellt werden. Visual Studio erlaubt es, Apps mit Webtechnologien zu erstellen. So können Apps mit Hypertext Markup Language (HTML), CSS und Javascript erstellt werden. Das Markup wird mit HTML erstellt, dieses dann mit CSS gestylt und die Programmlogik mit JavaScript realisiert. Die resultierende App ist trotzdem eine native Windows 8 Anwendung und kann nur auf Windows 8 oder Windows RT Geräten installiert und ausgeführt werden. Das Design und das Layout einer solchen App können in Visual Studio allerdings nicht visuell oder per Drag & Drop entworfen werden. Für diesen Zweck liefert Microsoft ein gesondertes Programm mit, *Blend für Visual Studio 2012*. In Blend können HTML und CSS Änderungen direkt in einer Live-Vorschau angesehen werden. Außerdem können hier auch Elemente per Drag & Drop angeordnet und verschoben werden. Ein geöffnetes Visual Studio Projekt kann mit zwei Klicks im Menü direkt in Blend geöffnet werden.

Auf diese Weise wird es Webentwicklern ermöglicht, einen komfortablen Einstieg in die Programmierung von Windows 8 Apps zu finden. Damit bleibt die Entwicklung nicht nur den .NET-Entwicklern vorbehalten. Ganz ohne Lernaufwand kommen allerdings auch die Webentwickler nicht aus. Um eine App mit Webtechnologien erstellen zu können, müssen sich die Entwickler zuvor mit der neuen API-Technologie

### *3. Konzeption der App*

vertraut machen (Bleske, 2012).

Das Hauptaugenmerk liegt hierbei auf Microsofts hauseigener JavaScript-Bibliothek *Windows Library for JavaScript (WinJS)*. Sie bietet viele wichtige Funktionen z.B. zur Handhabung von Steuerelementen oder zu Behandlung von großen Datenmengen.

Letztendlich macht es keinen großen Unterschied, ob mit herkömmlich Programmiersprachen oder mit Webtechnologien gearbeitet wird. Viele Apps lassen sich mit beiden Ansätzen gleich gut umsetzen. Da die Entwicklung von nativen Apps mit Webtechnologien ein neues und interessantes Konzept ist, soll es in dieser Arbeit ausprobiert und angewandt werden.

## 4. Umsetzung der App

Nachdem die Konzeption der App abgeschlossen ist, wird in dieser Sektion erläutert, wie die App umgesetzt wird. Dazu wird zunächst Microsofts Raster-Vorlage näher beleuchtet. Anschließend werden die Datenanbindung, die Darstellung der Daten sowie die Features erklärt. Am Ende dieser Sektion befindet sich die Entwicklungschronologie. Sie gibt einen detaillierten Einblick in den zeitlichen Ablauf des Entwicklungsprozesses.

### 4.1. Rastervorlage<sup>7</sup>

Beim Erstellen einer Windows 8 Modern UI App mit HTML und Javascript bietet Microsoft von Haus aus verschiedene Vorlagen für Visual Studio, welche schon einige Grundfunktionen besitzen und sich so als guter Startpunkt für eine weiterführende App eignen. Die Vorlage, die für diese App verwendet wird, ist die Raster-App (engl. Grid Application) Vorlage. Sie bietet die Grundarchitektur für ein hierarchisches Navigationssystem wie es in Sektion 3.3.2 auf Seite 14 beschrieben ist.

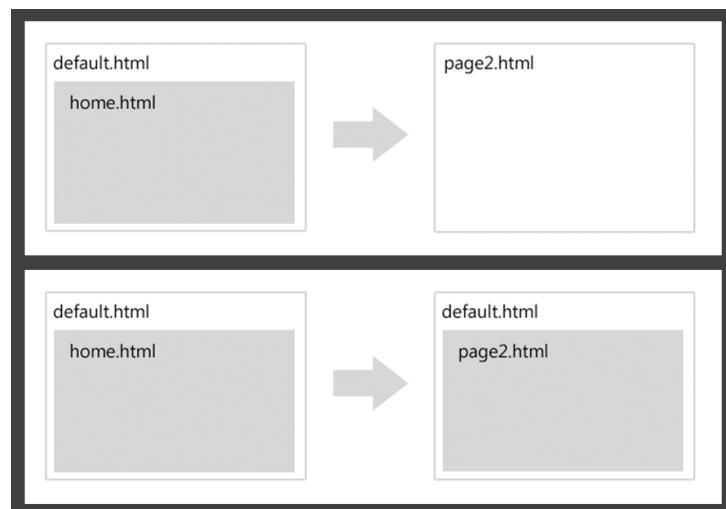


Abbildung 15: Oben: Die herkömmliche Navigation | Unten: die Single-Page-Navigation. (Microsoft, 2013b)

Es wird von Microsoft empfohlen, dass Windows-Store Apps, die mit HTML und Javascript erstellt werden, das single-page Navigationmodell verwenden. Die Navigation erfolgt nicht über Hyperlinks, sondern die verschiedenen Inhalte werden direkt in die Wrapper-Seite nachgeladen, ähnlich wie ein iframe. Abbildung 15 verdeutlicht das Single-Page-Navigation Konzept. Auf diese Weise gibt es keine sichtbare Unterbrechung beim Navigieren, d.h. es ist kein zwischenzeitlich weißer Bildschirm zu sehen. Außerdem erzielt man auf diese Weise eine bessere Performance und die App

<sup>7</sup>Die Namen der verwendeten Dateien in der Rastervorlage wurden mittlerweile von Microsoft geändert.

#### 4. Umsetzung der App

führt sich mehr „app-like“ an. Die Rastervorlage verwendet das single-page Navigationsmodell (Microsoft, 2013b). Abbildung 16 zeigt die Dateistruktur des Projektes. Anschließend wird erläutert, wofür die einzelnen Dateien verwendet werden.

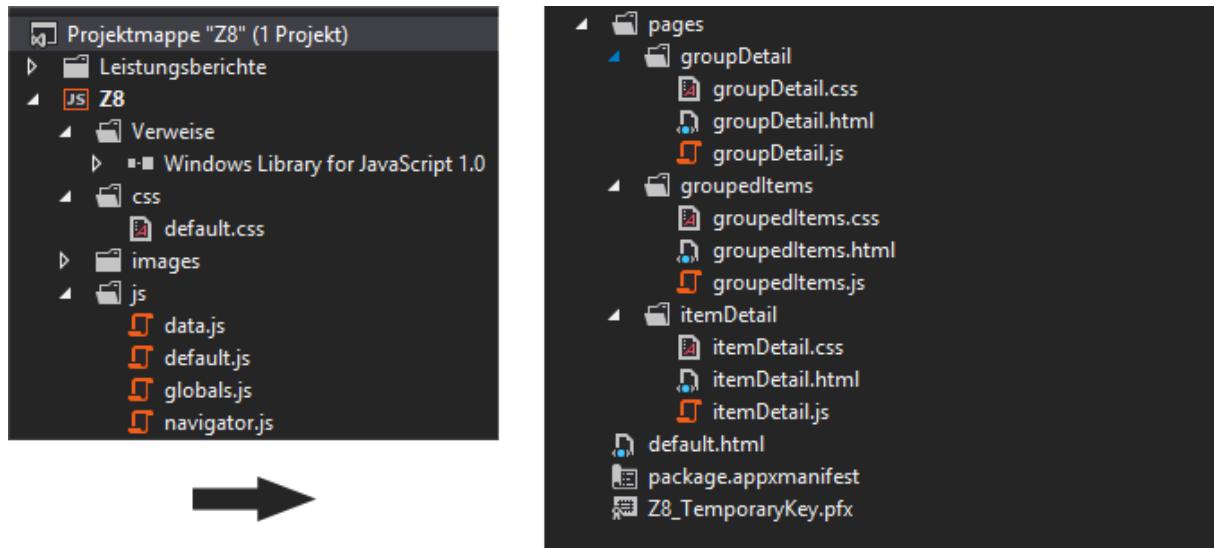


Abbildung 16: Die Projektstruktur in Visual Studio.

##### 4.1.1. HTML Dateien

Folgende HTML Dateien sind bereits in der Raster-App Vorlage enthalten:

- Die *default.html* Datei wird als erstes geladen und enthält das HTML für den Inhaltshost, dies ist die Seite, in welche die anderen Inhalte im Zuge der single-page Navigation herein geladen werden.
- Die *groupedItems.html* Datei ist der Einstiegspunkt in die App (die Hubansicht).
- Die *groupDetail.html* Datei zeigt alle Elemente eines Bereichs.
- Die *itemDetail.html* Datei beinhaltet die Einzelansicht eines Elements.

##### 4.1.2. Javascript Dateien

Folgende Javascript Dateien sind bereits in der Raster-App Vorlage enthalten:

- Die *default.js* Datei legt fest, wie sich die App beim Start verhält.
- Die *groupedItems.js*, *groupDetail.js* und *itemDetail.js* Dateien sind die Javascript Dateien, welche das Verhalten für die gleichnamigen HTML Dateien festlegen.
- Die *navigator.js* Datei implementiert das hierarchische Navigationssystem sowie das single-page Navigationmodell.
- Die *data.js* Datei stellt die benötigten Daten für den Rest der App bereit.

#### 4.1.3. CSS Dateien

Folgende CSS Dateien sind bereits in der Raster-App Vorlage enthalten:

- Die *default.css* Datei enthält die Styles für Inhaltshostseite sowie andere globale Styles.
- Die *groupedItems.css*, *groupDetail.css* und *itemDetail.css* Dateien enthalten die Styles für die gleichnamigen HTML Dateien.

Diese Vorlage wird nachfolgend angepasst und verändert, sodass sie die konzeptionierten Anforderungen erfüllt.

## 4.2. Datenanbindung

Damit Nachrichten in der App angezeigt werden können, muss eine Datenverbindung zu ZEIT ONLINE hergestellt werden. Diese Verbindung muss nicht permanent sein, es reicht wenn beim Start der App die aktuellen Daten eingelesen werden. Sollte der Bedarf nach weiteren Daten bestehen, kann jederzeit eine neue Anfrage abgesetzt werden. Zunächst wird erläutert wie der Datenbestand von ZEIT ONLINE generiert wird. Anschließend wird darauf eingegangen, wie diese Daten in der App verfügbar gemacht werden.

### 4.2.1. ZEIT ONLINE Datenbestand

Es soll zunächst geklärt werden, wie sich der Inhalt der ZEIT ONLINE Website darstellt und wie er generiert wird. Die redaktionellen Inhalte werden fast ausschließlich über ein hauseigenes Content Management System (CMS) erstellt und gepflegt. Das CMS generiert aus den Inhalten eine Extensible Markup Language (XML)-Struktur. Aus dem XML wiederum wird anschließend mit Hilfe von Extensible Stylesheet Language Transformations (XSLT) das fertige HTML erstellt.

Für die Windows 8 App wird direkt das XML verwendet, welches vom CMS generiert wird. Für die App werden zwei verschiedene Seitentypen der Webseite benötigt, um die gewünschten Informationen darzustellen, zum einen die verschiedenen „Centerpages“ und zum anderen die Artikelansicht. Centerpages sind die Hauptseiten der jeweiligen Ressorts (z.B. die Hauptseite des Politik-Ressorts) sowie die Startseite mit den wichtigsten und aktuellsten Meldungen. Auf den Centerpages befinden sich die Teaserelemente für die verschiedenen Artikel. Die Teaserelemente bestehen meist aus einem Bild und einem kurzen, prägnanten Text, der erläutert, worum es in dem dahinter liegenden Artikel geht.

```

1 <block href="http://xml.zeit.de/digital/internet/2013-08/fablab-open-
   hardware" year="2013" issue="34" ressort="Digital" author="Tilman Baumgä-
   rtel" contenttype="article" publication-date="" expires="" date-last-
   modified="2013-08-14T12:58:40+00:00" date-first-released="2013-08-14T09
   :57:43.627551+00:00" date-last-published="2013-08-14T12
   :59:39.691370+00:00" last-semantic-change="2013-08-14T09
   :56:40.185797+00:00">
2   <supertitle>Open Hardware</supertitle>
3   <title>Fab Labs, die Maschinen-Bibliotheken</title>
4   <text>
5       3-D-Drucker, CNC-Fräsen oder Lasercutter - mit solchen Maschinen
         sollen Bastler in Fab Labs experimentieren. Immer mehr solcher
         Werkstätten entstehen nun in aller Welt.
6   </text>
7   <description>
8       3-D-Drucker, CNC-Fräsen oder Lasercutter - mit solchen Maschinen
         sollen Bastler in Fab Labs experimentieren. Immer mehr solcher
         Werkstätten entstehen nun in aller Welt.
9   </description>
10  <byline/>
11  <image alt="MakerBot Replicator 2" align="left" title="MakerBot
           Replicator 2" base-id="http://xml.zeit.de/digital/internet/2013-08/
           makerbot-cebit-hannover/" type="jpg" publication-date="" expires="">
12    <bu>
13        Ein MakerBot Replicator 2 auf der diesjährigen Cebit in Hannover
14    </bu>
15    <copyright>© REUTERS/Fabrizio Bensch</copyright>
16  </image>
17 </block>
```

Listing 1: Das XML eines Teaserelements

Das XML von einem Teaserelement ist in Listing 1 dargestellt. Es werden jedoch nicht alle Informationen aus dem XML verwendet. Die verwendeten Informationen sind das „date-first-released“ (Zeile 1), der „title“ (Zeile 3), sowie die „description“ (Zeile 7) und das „image“ (Zeile 11). Das XML für die Artikelansicht ist ähnlich aufgebaut, nur ist hier zusätzlich noch der Artikeltext in Form von Paragraphen-Tags (`<p>`) enthalten. Wie die Daten im Detail verarbeitet werden, wird in der nächsten Sektion erläutert.

#### 4.2.2. Daten verfügbar machen

Die Daten, die in der App dargestellt werden sollen, werden grundsätzlich in zwei Schritten geladen. Wenn die App startet, werden zunächst alle benötigten Daten außer den eigentlichen Artikeln geladen. Der Artikelinhalt wird erst geladen, wenn der

User ihn lesen will. Dies spart Zeit beim Starten der App, und der User hat somit ein flüssigeres App-Erlebnis. Der erste Schritt beim Einlesen und Verarbeiten der Daten geschieht in der *data.js* Datei. Es werden alle Teaserelemente aller Ressorts letztendlich in einer *WinJS.Binding.List* gespeichert und für den weiteren Gebrauch verfügbar gemacht. Die WinJS ist eine Javascript Bibliothek für Windows-Store Apps, die mit Javascript erstellt werden. Sie enthält nützliche Funktionen z.B. für UI Steuerelemente oder sie hilft beim XMLHttpRequest (XHR). Die Binding.List ist ebenfalls ein Teil dieser Bibliothek. Sie stellt Logik für die Datengruppierung bereit, genau in der Art und Weise, wie es für diese App sinnvoll ist. Falls mit dynamischen Daten gearbeitet, wird stellt sie z.B. Funktionen für die automatische Aktualisierung der Daten bereit. In Listing 2 ist die Initialisierung einer *WinJS.Binding.List* dargestellt.

```
1 var teaserElements = new WinJS.Binding.List();
```

Listing 2: Initialisierung der Binding-List.

Die Grundinformationen zu den einzelnen Ressorts wie der Name und die URL werden zunächst als normales Javascript Array festgelegt. Einen Beispieleintrag aus dem Array zeigt Listing 3.

```
1 ressorts = [
2   {
3     key: "ressort01", url: http://xml.zeit.de/index,
4     title: Top Stories, subtitle: subtitle, updated: tbd,
5     backgroundImage: tbd, articleLink: "tdb",
6     acquireSyndication: acquireSyndication, dataPromise: null
7   }
8 ]
```

Listing 3: Auszug aus dem Ressorts Array.

Anschließend wird eine weitere Funktion von WinJS verwendet, die *WinJS.Promise*. Mit Promises kann mit asynchronen Prozessen und Datenquellen umgegangen werden. Hier werden für alle Ressorts XHRs gestartet, an die im Ressorts-Array angegebene URLs. Alle XHRs werden ebenfalls in einem Array gespeichert und erst wenn alle Promises vorhanden sind (alle URL waren erreichbar), wird die Datenverarbeitung fortgesetzt.

Mit den vorhandenen und validen XHR Responses können anschließend alle Teaserelemente in einer Schleife durchlaufen, das XML geparsst und so die nötigen Informationen für die App verfügbar gemacht werden. Listing 4 zeigt wie der Titel (Zeile 10), der TeaserText (Zeile 13) und der Bildpfad (Zeilen 15-22) aus dem XML gewonnen werden. Am Ende der Funktion werden die Daten in die in Listing 2 erstell-

#### 4. Umsetzung der App

te *WinJS.Binding.List* geschrieben (Zeilen 24-28). Die dargestellte Funktion ist nicht vollständig und wurde aus Übersichtlichkeitsaspekten und Relevanzgesichtspunkten verkürzt dargestellt.

```
1 function getItemsFromXml(ressortXML, teaserElements, ressort) {
2     var teasers = ressortXML.querySelectorAll("region[area=lead] container >
3         block:first-child");
4     // Process each ressort teaser.
5     for (var teaserIndex = 0; teaserIndex < teasers.length; teaserIndex++) {
6         var teaser = teasers[teaserIndex];
7
7         //only articles with an image are allowed
8         if (teaser.getAttribute("contenttype") == "article" && teaser.
9             querySelector("image") !== null) {
10            // Get the title
11            var teaserTitle = teaser.querySelector("title").textContent;
12
12            // Process the content so that it displays nicely.
13            var staticContent = toStaticHTML(teaser.querySelector("description
14                ").textContent);
14
15            //Get and build the image path
16            var teaserImageEl = teaser.querySelector("image");
17            var imagebasePath = teaserImageEl.getAttribute("base-id");
18            var splitImagePath = imagebasePath.split("/");
19            var imageNameSmall = splitImagePath[splitImagePath.length - 2] + "
20                -220x124.jpg";
20            var imageNameBig = splitImagePath[splitImagePath.length - 2] + "
21                -540x304.jpg";
21            var imagePathSmall = imagebasePath + imageNameSmall;
22            var imagePathBig = imagebasePath + imageNameBig;
22
23
24            // Store the teaser element info we care about in the array.
25            teaserElements.push({
26                group: ressort, key: ressort.title, title: teaserTitle,
27                backgroundImage: imagePathBig, teaserText: staticContent
28            });
29        }
30    }
31 }
```

Listing 4: Parsen und Speichern der Daten.

Sobald die Daten in der *WinJS.Binding.List* gespeichert sind, können sie vom Rest der App weiterverwendet und grafisch aufbereitet werden. Es wurden bisher noch keine Artikelinhalte geladen, dies geschieht erst dann, wenn der User auf einen Artikel klickt.

### 4.3. Darstellung der Daten

In dieser Sektion wird darauf eingegangen, wie die zuvor verarbeiteten und gespeicherten Daten für die ansprechende Darstellung in der App aufbereitet werden.

#### 4.3.1. Die Hubansicht

Wenn die App startet, befindet sich der User in der Hubansicht. Hier werden ihm für alle Haupt-Ressorts die ersten sechs Artikel angezeigt, in der Reihenfolge wie sie auch auf der Webseite zu finden sind (siehe Abbildung 17).

Zu der Hubansicht gehören eine HTML Datei, in welcher das Markup festgelegt ist, eine CSS Datei, die das Layout der Seite bestimmt und eine Javascript Datei, in der das Verhalten (die Logik) der Hubseite programmiert wird.

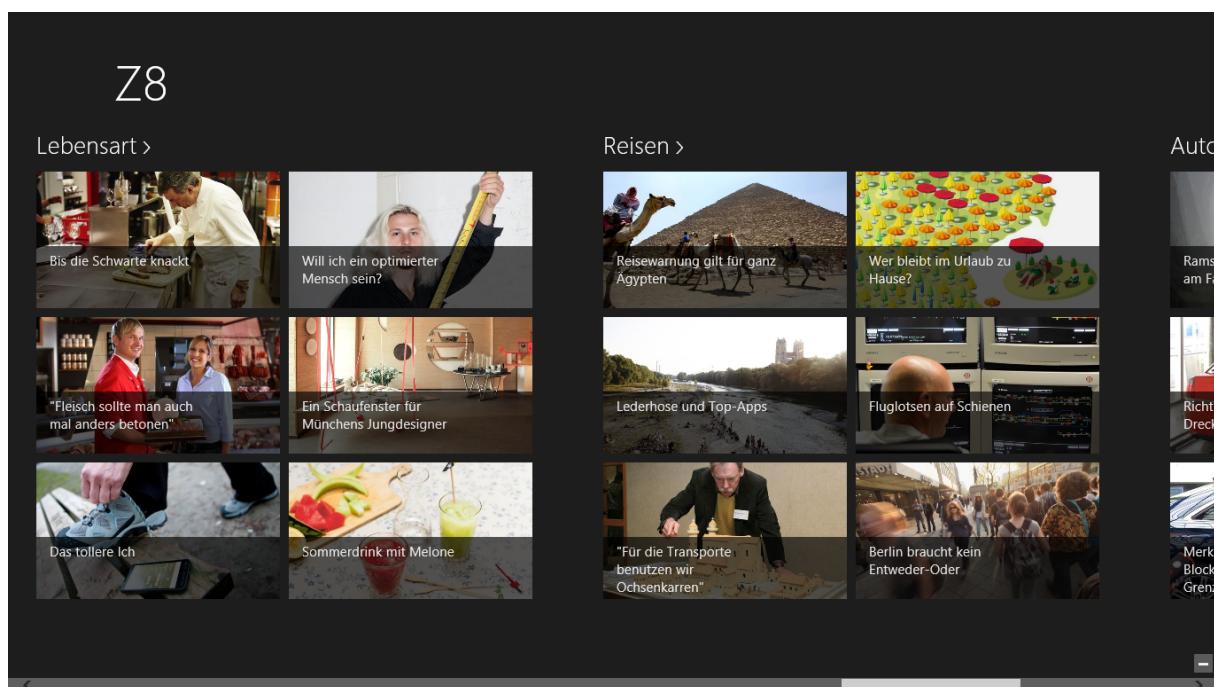


Abbildung 17: Die Hubansicht.

Um die Daten aus der `WinJS.Binding.List` auf der Hubseite anzeigen zu können, sind einige Schritte notwendig. Zunächst soll der Aufbau der HTML Datei erläutert werden. Listing 5 zeigt einen Ausschnitt aus der `groupedItems.html` Datei. Im oberen Teil befindet sich ein Template mit der Klasse `itemtemplate`, welches auf jedes einzelne Element der Hubansicht angewandt wird. Das besondere an diesem Template ist, dass es ein `data-win-control` Attribut mit dem Wert „`WinJS.Binding.Template`“ besitzt. Es akzeptiert Daten aus einer `WinJS.Binding.List`, wie sie in der Rohdatenverarbeitung genutzt wurde. Im unteren Teil ab Zeile 10 steht der Container für den eigentlichen Inhalt. Dieses Objekt besitzt ein `data-win-control` Attribut mit dem Wert „`WinJS.ListView`“ . Eine `WinJS.ListView` stellt Elemente in anpassbaren Listen oder Rastern dar.

```

1 <div class="itemtemplate" data-win-control="WinJS.Binding.Template">
2   <div class="item">
3     
5     <div class="item-overlay">
6       <h4 class="item-title" data-win-bind="textContent: title"></h4>
7     </div>
8   </div>
9
10 <section aria-label="Main content" role="main">
11   <div class="groupeditemslist win-selectionstylefilled" aria-label="List
12     of groups" data-win-control="WinJS.UI.ListView" data-win-options="{
      selectionMode: none}"></div>
13 </section>

```

Listing 5: Die wichtigsten Markupelemente der Hubansicht.

Der Rest passiert in der *groupedItems.js* Datei. Die *groupedItems.js* besitzt eine *ready()* Methode. Diese wird jedesmal aufgerufen, wenn der User zur Hubansicht wechselt oder die Hubansicht wiederhergestellt wird. Um die Daten in der *WinJS.ListView* zu hinterlegen wird, als erstes die *WinJS.ListView* in einer Variable gespeichert und anschließend das gewünschte Template bei der ListView registriert (siehe Listing 6).

```

1 var listView = element.querySelector(".groupeditemslist").winControl;
2 listView.itemTemplate = element.querySelector(".itemtemplate");

```

Listing 6: Template bei der Listview registrieren.

Anschließend wird die *initializeLayout* Funktion aufgerufen. Hier wird am Anfang eine entscheidende Operation ausgeführt. In der *WinJS.Binding.List* sind bisher alle Teaserelemente aus allen Ressorts enthalten, für die Hubansicht sind aber nur sechs Elemente pro Ressort gewünscht. Deswegen wird eine Funktion *getClippedList()* in der *data.js* Datei aufgerufen, welche die ersten sechs Elemente jedes Ressorts zurück liefert. Damit die Elemente wie gewünscht angezeigt werden können, muss noch die *dataSource* der *WinJS.ListView* gesetzt werden, ebenso wie das Raster-Layout (siehe Listing 7). Nach diesem Schritt werden die ersten sechs Elemente jedes Ressorts wie in Abbildung 17 angezeigt.

#### 4. Umsetzung der App

```
1 listView.itemDataSource = groupedItemsHub.dataSource;
2 listView.layout = new ui.GridLayout({ groupHeaderPosition: "top" });
```

Listing 7: Setzen der Datenquelle und des Layouts.

##### 4.3.2. Die Ressortansicht

Die Ressortansicht, die zweite Detailstufe zeigt alle Elemente eines Ressorts (siehe Abbildung 18). Der User gelangt hierhin, wenn er in der Hubansicht auf den Ressortnamen im Headerbereich klickt.

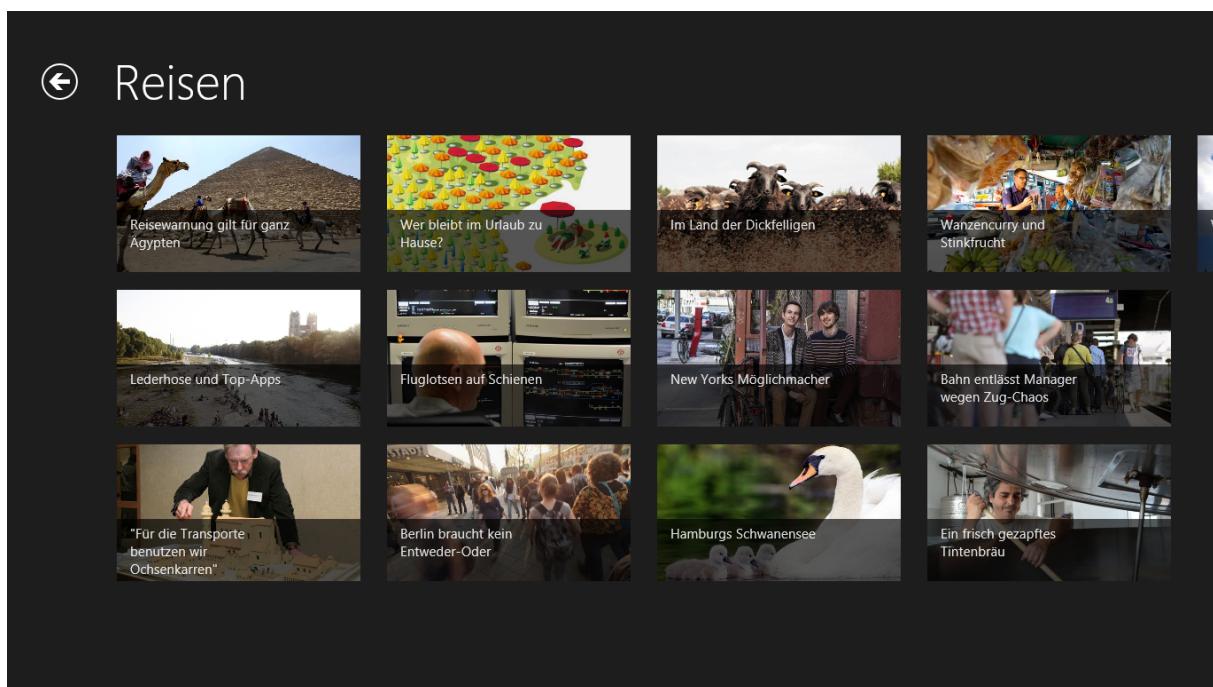


Abbildung 18: Die Einzelressortübersicht (hier das Reisen-Ressort).

Die Programmlogik funktioniert analog zur Hubansicht. Es werden die gleichen Mechanismen zur Darstellung des Raster-Musters benutzt. Auch die zu dieser Ansicht gehörende HTML und Javascript Dateien ähneln denen von der Hubansicht sehr. Ein wesentlicher Unterschied besteht darin, dass im Gegensatz zu Hubansicht nicht die Funktion `getClippedList()`, sondern stattdessen die Funktion `getItemsFromGroup()` aufgerufen wird. Sie gibt alle Elemente der übergebenen Gruppe zurück. Diese können nun mit einem `itemTemplate` und einer `WinJS.ListView` dargestellt werden.

##### 4.3.3. Die Artikelansicht

Die Artikelansicht ist die letzte Detailstufe. Sie zeigt den eigentlichen Artikelinhalt (siehe Abbildung 19). Dargestellt werden das Ressort, der Titel, der TeaserText, das

#### 4. Umsetzung der App

Aufmacherbild in groß, die Bildunterschrift, das Copyright für das Bild, der Artikeltext, die Quelle des Artikels (z.B. ZEIT ONLINE oder dpa), der Autor sowie das Erstellungsdatum. Die Artikeldarstellung funktioniert etwas anders als die Hub- und der Ressortdarstellung. Es wird kein Template verwendet, stattdessen wird das HTML mit Hilfe von Javascript befüllt. In der *ready()* Funktion der *itemDetail.js* Datei werden zunächst die bereits vorhandenen Informationen in das HTML eingefügt.



Abbildung 19: Die Artikelansicht.

Listing 8 zeigt das Einfügen von Titel, Teaser text und dem Bild. Anschließend fehlt noch der Artikeltext, die Quelle und der Autor. Um diese Daten zu bekommen und zu speichern, ist ein weiterer XHR nötig. Der XHR wird mit der Artikel-URL aufgerufen, die in jedem Teaser element gespeichert ist. Auf diese Weise bekommt man das XML der Artikel, wie es auch von der Webseite verwendet wird.

```
1 element.querySelector("article .item-title").textContent = item.title;
2 element.querySelector("article .item-subtitle").innerHTML= item.teaserText;
3 element.querySelector("article .item-image").src = item.backgroundImage;
```

Listing 8: Das Einfügen von Titel sowie Teaser text und Bild.

Beim Parsen des Artikel-XMLs müssen folgende drei Sachverhalte besonders beachtet werden. Die Absätze des Artikels liegen in Paragraphen-Tags (*<p></p>*) vor. Es kann vorkommen, dass außer dem reinen Text, zusätzlich noch Infoboxen im Artikel vorhanden sind. Der zweite Besonderheit ist die, dass Nachrichten vom Mikroblogging Dienst Twitter eingebunden werden können. Diese Abschnitte arbeiten ebenfalls mit Paragraphen-Tags. Da Infoboxen und Twitternachrichten sich inhaltlich nicht di-

rekt in den Artikeltext eingliedern, müssen diese Abschnitte vorher herausgefiltert werden. Des weiteren sind in den meisten Artikeln auf der ZEIT ONLINE Webseite Hyperlinks auf andere Artikel oder auf andere Webseiten vorhanden. Diese sind genau wie Infoboxen und Twitternachrichten unerwünscht, da sie, wenn sie vom User aufgerufen werden, außerhalb der aktuellen App im Browser geöffnet würden. Dies unterbricht und stört das App-Erlebnis. Es muss dementsprechend darauf geachtet werden, dass nur der reine Artikeltext als Artikel gespeichert wird. Listing 9 verdeutlicht, wie dies umgesetzt wird. In Zeile 3 werden die Infoboxen und die Twitternachrichten herausgefiltert und in den Zeilen 8-13 werden die `<a></a>` Tags mit Hilfe regulärer Ausdrücke gelöscht. Sobald der Artikel komplett geladen ist, wird er analog zu Listing 8 in das HTML Markup eingefügt.

```

1 for (var n = 0; n < paragraphs.length; n++) {
2     //exclude info box and tweets paragraphs
3     if (paragraphs[n].parentNode.parentNode.parentNode.parentNode.nodeName
4         != "infobox" && paragraphs[n].parentNode.getAttribute("class") != "
5             twitter-tweet")){
6         var xmlText = new XMLSerializer().serializeToString(paragraphs[n]);
7         if (paragraphs[n].querySelector("a") != null) {
8             var numberOfLinks = paragraphs[n].querySelectorAll("a").length;
9             //remove hyperlinks
10            for (var i = 0; i < numberOfLinks; i++) {
11                var rx = new RegExp("<a[^>]+>", "i");
12                xmlText = xmlText.replace(rx, "");
13                rx = new RegExp("</a>", "i");
14                xmlText = xmlText.replace(rx, "");
15            }
16        }
17        textContent = textContent + xmlText;
18    }
19 }
```

Listing 9: Vermeidung von Infoboxen und Twitternachrichten sowie Hyperlinks in den Artikeln.

#### 4.4. Features

Im Folgenden werden hier die Features der App erläutert. Es handelt sich hierbei zum einen um die Einstellungen, die der User über die untere Menüleiste vornehmen kann und zum anderen um das „Semantic Zoom“ Feature.

### 4.4.1. Artikeltitel ausblenden

Ein Hauptfeature der App ist es, dass in der Hubansicht und in der Ressortansicht die Artikeltitel ausgeblendet werden können und der User so nur noch eine Bilderwand vor sich hat. So kann er die Nachrichtenlage rein visuell erleben, und erst wenn er auf einen Artikel klickt, bekommt er im Artikel den Titel und Artikelinhalt zu sehen. Diese Funktion wird über eine Menüleiste am unteren Bildrand implementiert (siehe Abbildung 20). Um eine solche Menüleiste zu erzeugen, genügt ein `<div>` Container in der `default.html` Datei mit einem `data-win-control` Attribut, das den Wert „WinJS.UI.AppBar“ bekommt. In diesem Element können anschließend `<button>` Elemente angelegt werden. Diese sind später die sichtbaren Schaltflächen, wenn die Menüleiste aufgerufen wird.

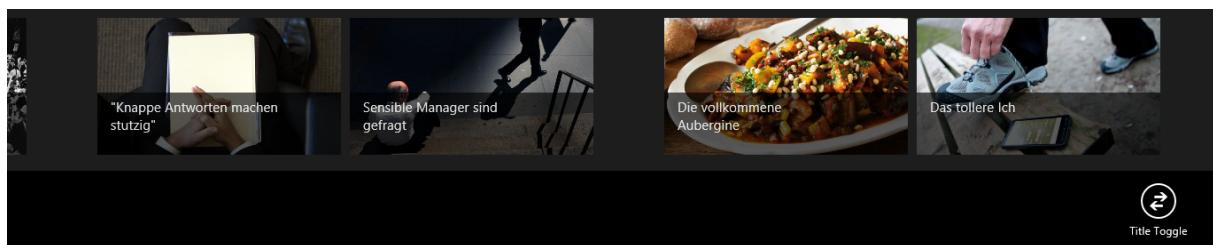


Abbildung 20: Die untere Menüleiste mit der Funktion zum Artikeltitel ausschalten.

In der Initialisierungsphase der App wird in der `default.js` Datei der Click-Handler für den Button registriert. Hier wird die Funktion angegeben, die aufgerufen werden soll, wenn der Button geklickt wird. In diesem Fall wird die Funktion `titleToggle()` in der `globals.js` Datei aufgerufen. In der `globals.js` Datei befinden sich Variablen und Funktionen, die von mehreren Ansichten benutzt werden und deswegen global verfügbar sein sollen. Die `titleToggle()` Funktion ist schließlich verantwortlich für das Ausblenden der Artikeltitel.

Die Funktion sammelt dazu alle Elemente mit der Klasse `item-overlay` in einem Array und setzt anschließend für alle gefundenen Elemente die CSS Eigenschaft `display` auf „none“ . Das Resultat ist in Abbildung 21 zu sehen.

### 4.4.2. Schriftgröße verändern

In der Artikelansicht hat der User die Möglichkeit, die Schriftgröße anzupassen. Es sind jeweils zwei Stufen größer als auch zwei Stufen kleiner als die normale Größe möglich. Der Ansatz ist grundsätzlich genau so wie bei der „Titel-Ausblenden“ Funktionalität. Zunächst wird ein Button in der unteren Menüleiste angelegt. Es wäre auch möglich, zwei Buttons zu erstellen, einen für größer und einen für kleiner. Um weitere Entwicklungsmöglichkeiten der WinJS-Bibliothek auszunutzen, gibt es in der App nur einen Button in der Menüleiste. Wird dieser geklickt, erscheint noch ein weiteres Menü, ein sogenanntes *Flyout-Menü*. In diesem sind schließlich die Schaltflächen für „Schrift größer“, „Schrift kleiner“ und für „Schriftgröße zurücksetzen“ hinterlegt. Ist

#### 4. Umsetzung der App

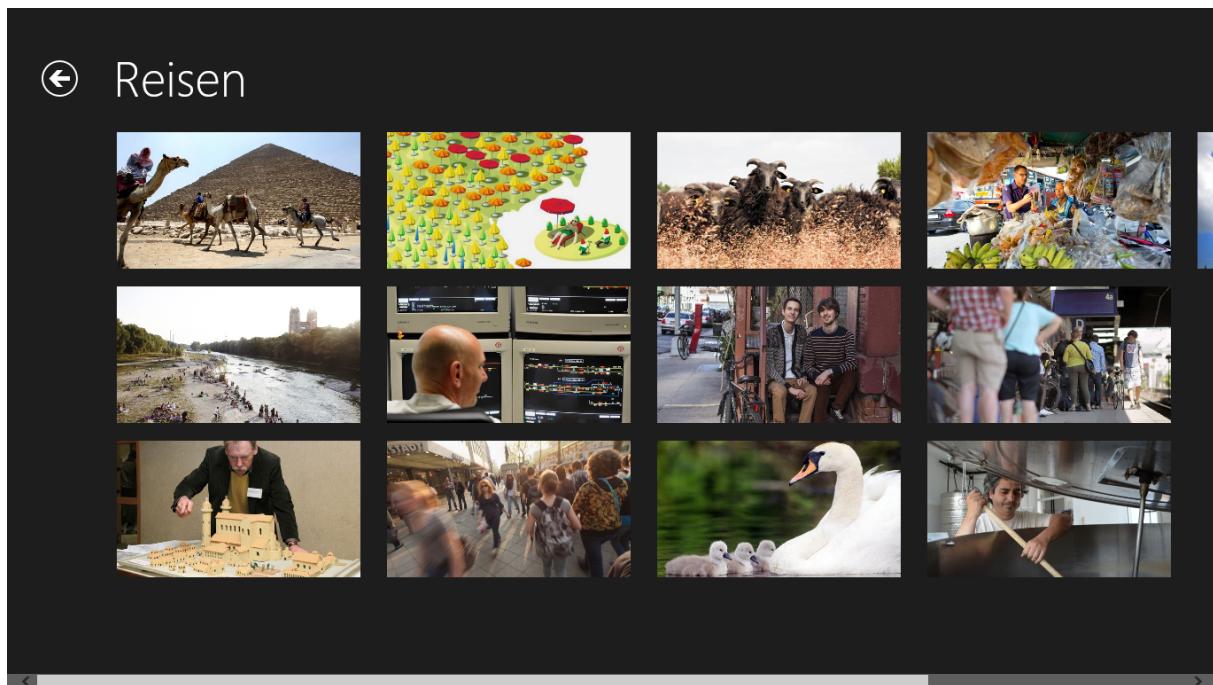


Abbildung 21: Das Reisen-Ressort ohne Artikeltitel.

die Schriftgröße z.B. auf der größten Stufe eingestellt, wird im Menü der „Größer Button“ nicht mehr dargestellt. Abbildung 22 zeigt die verschiedenen Stati, die das Schriftgrößenmenü haben kann.

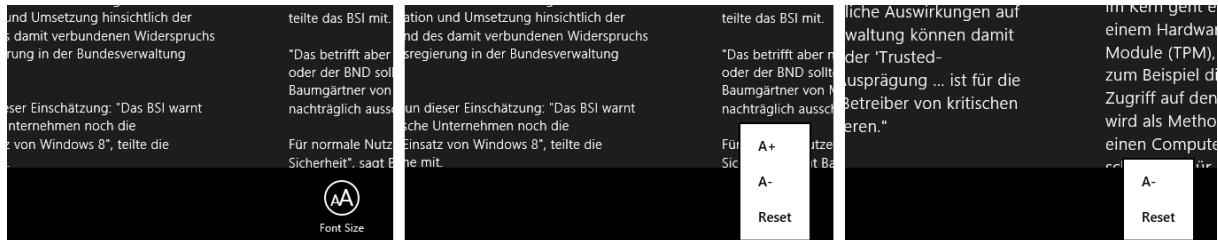


Abbildung 22: Links: Schriftgröße Button im unteren Menü | Mitte: Das Flyout-Menü mit allen Schaltflächen. | Rechts: Das Flyout-Menü beim Aufruf wenn der größte Wert eingestellt ist.

#### 4.4.3. Semantic Zoom

Das „Semantic Zoom“ Feature wird eingesetzt, wenn es in der Hubansicht sehr viele Bereiche gibt. Es soll dem User ermöglichen ohne langes Scrollen, zu den verschiedenen Bereichen in der Hubansicht zu springen. Beim Aufruf des „Semantic Zooms“ wird eine weitere Ebene über die Hubansicht gelegt. Typischer Weise werden in dieser Ebene die verschiedenen Bereiche ohne inhaltliche Elemente angezeigt. Bei ZEIGHT werden hier alle Ressorts mit dem jeweiligen Ressortnamen und einem zufälligen Bild aus dem Ressort angezeigt (siehe Abbildung 23). Diese Ansicht ist ebenfalls horizontal scrollbar, im direkten Vergleich zur Hubansicht jedoch viel kompakter und übersichtlicher. Klickt der User z.B. auf das Reisen-Ressort, wird wieder zurück zur

#### 4. Umsetzung der App

Hubansicht „gezoomt“. Der User hätte dann die Ansicht, wie sie Abbildung 17 auf Seite 25 zeigt. Damit findet in beide Richtungen ein kontextsensitives Zoomen statt. Je nach dem wo sich der User in der Hubansicht befindet bzw. auf welches Ressort der User in der „Semantic Zoom“ Ebene klickt, wird an die entsprechende Stelle raus- bzw. reingezoomt.

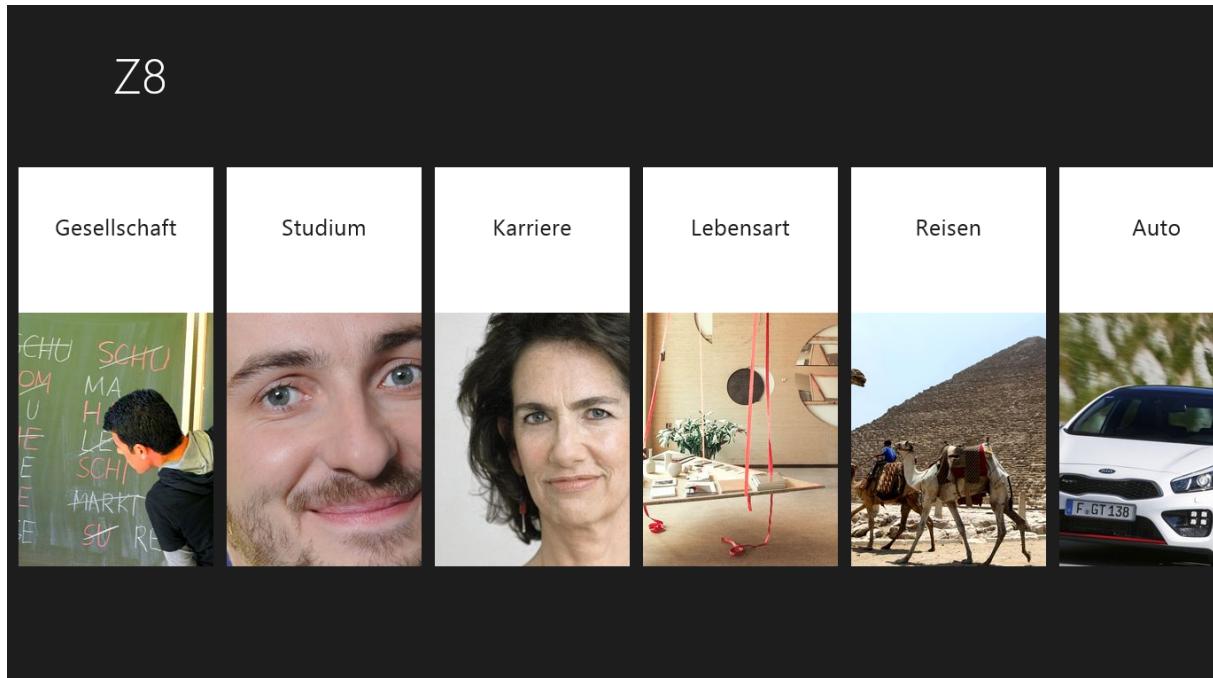


Abbildung 23: Die „Semantic Zoom“ Ansicht.

Da der semantic Zoom von der Hubansicht aus aufgerufen wird und mit ihr interagiert, ist er dementsprechend in der `groupedItems.html` und in der `groupedItems.js` implementiert. Genau wie für die Hubansicht gibt es auch für den semantic Zoom ein Template für die einzelnen Elemente. Damit eine Interaktion zwischen der Hubansicht und dem semantic Zoom stattfinden kann, wird ein Wrapper `div` Element um die `WinJS.ListView` der Hubansicht und die `WinJS.ListView` der „Semantic Zoom“ Ebene gezogen. Listing 10 zeigt, wie der Wrapper eingebunden ist. Das `div` Element besitzt ein `data-win-control` Attribut mit dem Wert „`WinJS.UI.SemanticZoom`“. Des Weiteren kann der `zoomFactor` und der initiale Zoomstatus festgelegt werden. Bei der ZEIGHT App steht dieser Zoomstatus initial auf „false“, da der User die Hubansicht als Einstiegspunkt in die App haben soll und nicht die „Semantic Zoom“ Ansicht.

```

1 <div class="semanticZoom" data-win-control="WinJS.UI.SemanticZoom" data-win-
   -options="{ zoomFactor: 0.5, initiallyZoomedOut: false }" style="height:
     100%>
2   <div class="groupeditemslist win-selectionstylefilled" aria-label="List
      of groups" data-win-control="WinJS.UI.ListView" data-win-options="{
        selectionMode: none}"></div>
3   <div class="groupeditemslistZoomOut groupeditemslist" aria-label="List
      of groups" data-win-control="WinJS.UI.ListView" data-win-options="{
        selectionMode: none}"></div>
4 </div>
```

Listing 10: Der Semantic Zoom Wrapper.

## 4.5. Entwicklungschronologie

Hier soll beschrieben werden, wie der Entwicklungsprozess von ZEIGHT abgelaufen ist. Nachdem die Idee für die App formuliert war, hat sich der Verfasser zunächst mit der Windows 8 App Entwicklung im Allgemeinen vertraut gemacht, da bis zu diesem Zeitpunkt keinerlei Windows 8 spezifische Entwicklungskenntnisse vorhanden waren. Es wurden einige Beispiele ausprobiert, um ein Gefühl für die Entwicklungs-umgebung zu bekommen. Der Workflow zwischen Visual Studio und Blend war am Anfang noch etwas gewöhnungsbedürftig. Anschließend wurde ein erster Prototyp erstellt. Hier fanden noch keine Überlegungen zum Navigationssystem oder zur Usability statt. Am Anfang gab es zunächst nur das Ziel, die ZEIT ONLINE Daten in irgendeiner Weise in die App zu übernehmen und sie dort darzustellen. In einer der ersten Versionen der App wurden z.B. nur die Artikeltitel der Homepage angezeigt. Nachdem die Datenanbindung umgesetzt war, konnte mit dem Entwickeln der richtigen App begonnen werden.

Es wurde zunächst viel recherchiert, zu nennen sind hier insbesondere die Microsoft Richtlinien und Styleguides, um herauszufinden, wie aus Microsoft Sichtweise eine Nachrichten-App auszusehen hat. Dabei war immer wieder die Rede von der Rastervorlage, die sich für solche Zwecke eignen solle. Diese Grundvorlage wurde schließlich für die App ausgewählt. Da die Rastervorlage keine triviale Vorlage ist, musste einige Zeit investiert werden, um alle Vorgänge und Dateien der Vorlage richtig ein- und zuordnen zu können.

Eines der ersten größeren Probleme war die Dezimierung der Daten für die Hubansicht. Hier musste die Funktion *getClippedList* entwickelt werden (wie in Sektion 4.3.1 beschrieben). Des weiteren wurde in dieser Phase das CSS für Hubansicht und die Ressortübersicht angepasst. Gerade bei der Ressortansicht musste einiges an der Vorlage verändert werden.

#### 4. Umsetzung der App

Danach wurden die Menüleisten hinzugefügt und die *titleToggle* zum Ausschalten der Artikeltitel programmiert. Da bisher noch keine Versionierung der App vorhanden war, wurde zu diesem Zeitpunkt ein GitHub Repository angelegt<sup>8</sup> und die App unter Versionskontrolle gestellt.

Anschließend wurde die Artikelansicht gestaltet und dabei die Bildunterschriften und die Copyrights positioniert. Leider kann bei Entwicklung mit HTML5/JavaScript nicht direkt auf die Scrollposition der Hubansicht zugegriffen werden. Deswegen landete der User beim Wechseln in die Hubansicht immer wieder ganz am Anfang. Da dies nicht zufriedenstellend war, wurde die Funktionalität manuell nachgerüstet, indem der Index des ersten sichtbaren Elements global gespeichert wird und beim Aufrufen der Hubansicht wieder gesetzt wird. Eine weitere manuell nachgerüstete Funktionalität ist der *loading Ring* beim Laden der App. Hierfür wird auf die Events der *WinJS.ListView* zurückgegriffen und die Animation ausgeblendet, sobald die ListView geladen ist.

Anschließend wurde die Verarbeitung der Artikel vereinfacht. Am Anfang wurden die *< p >* *< /p >* Elemente erst „auseinander genommen“ und anschließend wieder zusammengesetzt. Da das XML der Artikel aber schon valides HTML enthält, konnte einfach der ganze Text samt Zwischenüberschriften übernommen werden. Was noch herausgefiltert werden musste, sind Infoboxen und Twitternachrichten, die in den Artikeln eingebaut sein können. Als nächstes wurde das *Semantic Zoom* Feature sowie die Funktionalität zur Veränderung der Schriftgröße implementiert.

Zu diesem Zeitpunkt hat es eine kleine strukturelle Ergänzung im XML von ZEIT ONLINE gegeben. Die Teaserelemente haben nun jeweils ein Attribut *contenttype*. Da in der App nur Artikel dargestellt werden sollen, kann auf diese Weise bequem nach „*contenttype = article*“ gefiltert werden. Dies ist eine große Vereinfachung, denn zuvor wurden nach dem Negativ-Prinzip recht aufwändig alle anderen Typen wie z.B. Videos, Blogartikel oder Quizze ausgeschlossen.

Gegen Ende der Entwicklung wurde noch der Snapped Modus der App unterbunden. Normalerweise können Windows 8 Apps außer dem Vollbildmodus auch noch im Snapped-Modus ausgeführt werden. Dabei wird der Bildschirm vertikal getrennt, und es können zwei Apps im Verhältnis 70:30 nebeneinander dargestellt werden. Da der Snapped-Modus für diese App nicht relevant ist, wurde er unterbunden, ein vollständiges Abschalten ist nicht möglich. Zuletzt wurden noch Splashscreen und Logos in verschiedenen Größen hinzugefügt.

Es sind hier nur die wichtigen Schritte im Entwicklungsprozess beschrieben. Auf die Dokumentation der kleineren Tweaks und Fixes an der App wurde verzichtet. Des Weiteren sind auch nicht alle aufgetretenen Probleme erwähnt.

---

<sup>8</sup>Link zum GitHub Repository: <https://github.com/mohdr0w/Z8>

## 5. Evaluierung

Bei der Evaluierung soll herausgefunden werden, inwieweit die App intuitiv bedienbar ist und ob eine rein bildliche Darstellung ohne Artikeltitel beim Konsumieren von Nachrichten Sinn ergibt. Es wird hierzu eine Abwandlung der Fokusgruppenevaluierung angewendet.

### 5.1. Fokusgruppe

Im Gegensatz zu üblichen Fokusgruppen findet keine Diskussion der Teilnehmer untereinander statt, um unabhängige Meinungen zu bekommen. Aus der Theorie der Fokusgruppen wird der Aspekt der Heterogenität der Teilnehmer übernommen. Als Ergebnis werden aus den Meinungen der Teilnehmer Trendaussagen zu den Fragen der Evaluierung abgeleitet. Jeder Teilnehmer führt hierzu einige Aufgaben durch und stellt seine Meinungen und Anmerkungen anschließend im Interview dar.

Testperson (TP)	Beruf	Geschlecht	Alter
1	Business Coach	W	58
2	Leiter Finanz- und Rechnungswesen	M	58
3	Student - Games Master	M	27
4	Wirtschaftspsychologin	W	27
5	Student - Games Master	M	27

Tabelle 1: Übersicht der Testpersonen.

Tabelle 1 zeigt die Übersicht der Testpersonen. Es wurden Teilnehmer aus zwei Generationen ausgewählt, die sich von Beruf und Geschlecht unterschieden, um somit möglicherweise verschiedene Auffassungen und Herangehensweisen zu beobachten.

### 5.2. Ablauf

Der Ablauf der Evaluierung setzt sich aus drei Abschnitten zusammen. Im ersten Teil soll überprüft werden inwieweit die Testpersonen, allein vom Betrachten des Teaserbildes eines Artikels, auf den groben, oder ggf. auch auf den genauen Inhalt des Artikels schließen können. Hierzu werden den Testpersonen zwei Ressorts in der Hubansicht, bei ausgeschalteten Titeln, gezeigt. Zum einen das Politik-Ressort mit vorwiegend tagesaktuellen Nachrichten und zum anderen das Wissen-Ressort, welches auch zeitlose Artikel beinhalten kann. Abbildung 24 zeigt die Bilder, welche die Testpersonen beurteilen sollten.

Die Bilder sind nicht speziell zu diesem Zweck ausgewählt worden. Es handelt sich um die ersten sechs Artikel des jeweiligen Ressorts, wie sie am Sonntag den 25. August 2013 auch auf der ZEIT ONLINE Webseite zu finden waren. Die Testpersonen

## 5. Evaluierung

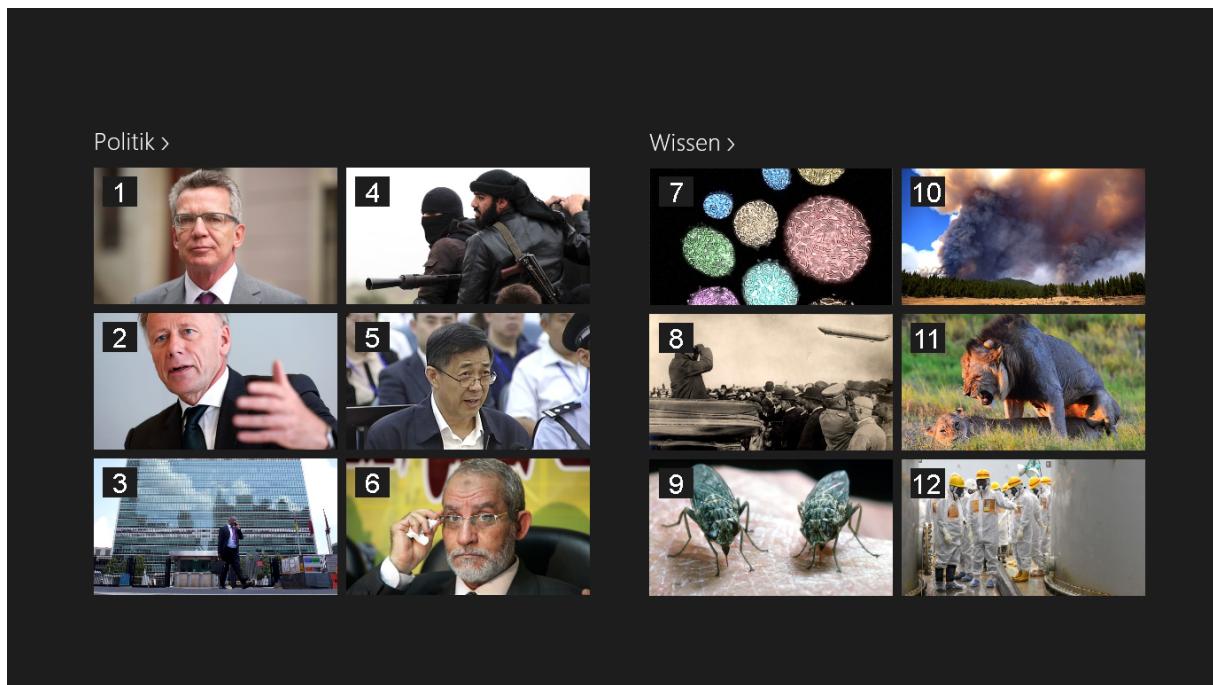


Abbildung 24: Die Testbilder aus dem Politik- und Wissen-Ressort.

sollen jedes Bild einzeln betrachten, zunächst eine Einschätzung zum groben Themengebiet geben und anschließend so genau wie möglich beschreiben, worum es in dem jeweiligen Artikel gehen könnte. Die Ergebnisse werden in einem Tabellenkalkulationsprogramm festgehalten.

Im zweiten Teil der Evaluierung sollen die Testpersonen die App selber benutzen. Zum einen auf einem Notebook, auf welchem die App mit der Maus bedient wird, und zum anderen auf dem *Microsoft Surface*, ein Tablet, welches mit Touchgesten bedient wird. Es gibt vorher keine Instruktionen, wie die App zu bedienen ist. Die Testpersonen sollen die App selbst entdecken und mögliche Features, die programmiert wurden, finden und ausprobieren. Dies soll zeigen, wie intuitiv das Navigationskonzept und die Menüführung der App sind.

Die Evaluierung endet mit dem dritten Teil, einem Interview. Hier sollen im Gespräch einige Fragen, zu den verschiedenen Teilen der Evaluierung beantwortet werden. Nachfolgend sind die Fragen aufgelistet, die im Einzelfall durch Nachfragen variiert werden können. Um den Redefluss der Teilnehmer nicht zu unterbrechen, werden die Interviews als Audiomitschnitt gespeichert und später ausgewertet. Quelle der aufgeführten Zitate sind diese Mitschnitte.

1. Kannst du kurz deine bisherigen Erfahrungen mit Windows 8 schildern?
2. In welcher Weise und wie oft konsumierst du Nachrichten im Allgemeinen?
3. Zu Teil 1: Wie war dein Gedankengang, um herauszufinden, was sich hinter den Bildern verbergen könnte?

## 5. Evaluierung

4. Zu Teil 2: Wie kamst du mit der App im Allgemeinen zurecht? War sie intuitiv? Welches Gerät hat dir besser gefallen und warum?
5. Glaubst du der „Nur-Bilder-Modus“ macht neugierig? Oder glaubst du er hemmt den Nachrichtenkonsum?

### 5.3. Ergebnisse

Hier sollen die Ergebnisse der Evaluierung dargestellt und bewertet werden. Zunächst geht es um die App und ihr Benutzungs- bzw. Navigationskonzept. Anschließend wird der Ansatz untersucht, ob Nachrichten auch auf eine andere Art und Weise konsumiert und entdeckt werden können.

#### 5.3.1. Intuitivität der App

Ob die App für die einzelne Testperson intuitiv bedienbar ist, hängt mit bereits gesammelten Windows 8 Erfahrungen zusammen. TP3-TP5 gaben bei der Frage nach der bisherigen Windows 8 Erfahrung an, bisher nur im Desktop-Modus von Windows 8 gearbeitet zu haben. Dies geschah ausschließlich mit Maus und Tastatur. TP1 hat bereits erste Erfahrungen mit der Modern UI Oberfläche gemacht. Außerdem ist sie vertraut mit dem Smartphonebetriebssystem von Microsoft, Windows Phone. TP2 hat bereits etwas tiefer gehende Erfahrungen mit Windows 8 und seinen Gesten, wenn auch hauptsächlich mit der Maus und weniger mit Touchbedienung.

TP1 fand das Navigationskonzept „absolut intuitiv“, wobei sie anmerkte, ihr hätte eine weitere Verbindung aus der Artikelansicht direkt zur Hubansicht gefallen. Klickt der User erst auf die Ressortübersicht und anschließend auf einen Artikel, kann nicht direkt zurück zur Hubansicht navigiert werden, sondern es muss der „Umweg“ zurück über die Ressortübersicht genommen werden. TP2 meinte die App sei gut strukturiert, da diese Ebenenstruktur bekannt sei, merkte aber gleichzeitig an, dass er sich vorstellen kann, das *nicht* Windows 8 Nutzer zunächst Schwierigkeiten mit der Bedienung und Gesten haben könnten. TP3 und TP4 fanden es etwas verwirrend, dass im Artikel horizontal gescrollt wird, im Gegensatz zu normalen Webseiten. TP5 „fehlte die nötige Vertrautheit mit der Modern UI Oberfläche“, um die nicht sofort sichtbaren Menüs aufzurufen. Die Texte hingegen seien sehr gut lesbar.

Generell lässt sich sagen, dass die Hauptnavigationsstruktur (Hubansicht, Ressortübersicht, Artikelansicht) von allen Testpersonen für intuitiv und gut verständlich empfunden wurde. Die meisten haben es auch geschafft, die Menüs mit den Features „Titel ausblenden“ und „Schriftgröße“ aufzurufen. Ein Feature wurde allerdings nur von einer Testperson gefunden, der „Semantic Zoom“. Wenn dieses Navigations-element von einer App benutzt wird, sollte gezielt darauf hingewiesen werden, da die Möglichkeit eines „Semantic Zooms“, gerade für Windows 8 Einsteiger, oft nicht intuitiv ersichtlich ist.

### 5.3.2. Der Nur-Bilder-Modus

Hier sollen die Möglichkeiten und die Sinnhaftigkeit des „Nur-Bilder-Modus“ dargestellt und untersucht werden. Alle Testpersonen gaben an, täglich über verschiedenste Kanäle Nachrichten zu konsumieren. Sei es im Fernsehen, Online, Radio, Smartphone oder die Zeitung. Das bedeutet, es ist zu vermuten, dass alle Testpersonen über ein Basiswissen, was tagesaktuelle Geschehnisse betrifft, verfügen.

Im ersten Teil der Evaluation sollten die Testpersonen versuchen herauszufinden, welcher Inhalt sich hinter den Teaserbildern von Abbildung 24 verbergen könnte. Hierzu sollte zunächst ein grobes Themengebiet genannt werden und anschließend so genau wie möglich versucht werden, zu bestimmen, worum es in dem jeweiligen Artikel geht. Ein Bild wurde in diesem Fall als „genau“ akzeptiert, wenn die Testperson nach dem obersten Themengebiet noch eine Stufe tiefer gehen konnte. So wurde es z.B. als richtig akzeptiert, wenn die Testperson bei Bild 10 nicht nur das grobe Thema „Waldbrände“, sondern auch „San Francisco“ oder „Yosemite Nationalpark“ richtig benannt hatte. Nach diesem Bewertungsschema wurden von den 12 Bildern im Durchschnitt 78% grob und 35% auch genauer richtig zugeordnet. Das am häufigsten richtig zugeordnete Bild ist Bild 10. Es wurde von allen Testpersonen genau richtig erkannt. Dicht gefolgt von Bild 5, welches von 4 Testpersonen genauer richtig zugeordnet werden konnte und von einer Testperson nur grob. Am wenigsten wurde Bild 3 erkannt. Es wurde nur von einer Testperson erkannt, dafür aber auch genau. Es geht in diesem Bild um die NSA-Affäre und den UN-Sicherheitsrat. Eine Übersicht mit allen zugehörigen Artikel Schlagzeilen befindet sich im Anhang A.

Die Testpersonen wurden im Interview gefragt, wie sie an diese Aufgabe herangegangen sind. In einem Punkt waren sich alle Testpersonen einig. Wenn nur eine Person abgebildet ist, müssen weitere Informationen über andere Nachrichtenkanäle bekannt sein, um den Inhalt genauer zu erkennen. TP1 meint, bei Bild 1 sei es sehr schwer, etwas genaues sagen zu können, da diese Person zu verschiedenen Themen in Frage kommt, z.B. Euro Hawk, Afghanistan, Syrien oder Strukturänderung Bundeswehr. TP5 sagt zu Bild 6: „Bei Bildern auf denen keine öffentlichen Personen drauf waren, hab ich einfach versucht über Vorurteile die Personen zu lesen“.

Bei den Bildern aus dem Politik-Ressort wurde sich sehr auf das bereits aus anderen Nachrichtenquellen Gehörte verlassen, wogegen bei den eher zeitlosen Artikeln und Bildern wie z.B. Bild 8 und Bild 11 die grobe Zuordnung recht leicht fiel, aber das Bild für die genaue Bestimmung nicht genügend Informationen lieferte und die Testpersonen auch nicht mit tagesaktuellerem Wissen auf die konkreten Inhalte schließen konnten. Bei Bild 8 geht es z.B. darum, dass Graf Zeppelin angeblich gar nicht der Erfinder des Zeppelins war und bei Bild 11 lautet der Titel: „Hat nur der Mensch Sex in der Missionarsstellung?“.

Es bleibt noch zu klären ob ein „Nur-Bilder-Modus“ sinnvoll ist und inwieweit der User dadurch neugierig gemacht wird oder ob dieser Modus ihn eher in seinem Nach-

## 5. Evaluierung

richtenkonsumfluss stört und hemmt. TP1 sagt dazu: „Das kommt drauf an, was ich gerade möchte, [...] wenn ich mich kurz und prägnant informieren möchte [...] muss unbedingt der Text drin sein, [aber] wenn ich mal ein bisschen Zeit habe und sitze irgendwo und warte, dann ist es eher der Entdeckermodus“. TP2 meint, es hänge von der Veranlagung der Person ab. Es sei eine offene Herangehensweise nötig: „Ich gucke mal was mir angeboten wird“. Weiterhin glaubt TP2, dass Inhalte so eventuell besser im Gedächtnis bleiben, weil der User auf diese Weise aktiv auf ein Bild, welches seine Neugier geweckt hat, klickt und sich damit den Inhalt gewissermaßen selbst „erarbeitet“ hat. TP4 sieht es ähnlich wie TP1. Es komme auf die Situation an und das Ziel des Nachrichtenkonsums an. TP5 meint, es hänge auch sehr stark vom jeweiligen Bild ab: „Es muss eine klare Verbindung zum Inhalt geben [,um die Neugier zu wecken,] was bei reinen Personenbildern nicht der Fall ist.“.

Die Testpersonen stimmen bei dieser Frage grundlegend überein. Wenn der User gezielt tagesaktuelle Nachrichten konsumieren will, ist ein „Nur-Bilder-Modus“ nicht zielführend. Es muss zu viel geraten und vermutet werden und im Zweifel klickt der User auf ein Bild und hat etwas ganz anderes erwartet und hat auf diese Weise ein Negativ-Erlebnis. Hat der User andererseits ein bisschen mehr Zeit und möchte mal etwas Neues ausprobieren, sich durch die Nachrichtenwelt treiben lassen, dann stellt der „Nur-Bilder-Modus“ eine interessante Alternative dar. Hier kann sich der User ganz auf seinen Neugier- und Entdecker-trieb verlassen und das anklicken, wo er vielleicht schon etwas vermutet und er wissen möchte ob er richtig liegt. Der Trend zeigt deutlich, um den „Nur-Bilder-Modus“ genießen zu können, sollte der User etwas Zeit mitbringen und sich bewusst darauf einlassen, nicht auf den ersten Blick alle Informationen dargelegt zu bekommen, sondern sich von der Neugier leiten zu lassen und es aktiv herauszufinden.

Zusammenfassend lassen sind folgende Punkte zum „Nur-Bilder-Modus“ feststellen. Um aus den Bildern auf Inhalte schließen zu können müssen Basiskenntnisse aus anderen Nachrichtenquellen vorhanden sein. Auf die groben Inhalte der Artikel kann bei den meisten Bildern geschlossen werden sowohl bei den aktuellen als auch bei zeitlosen Artikeln. Bei Bildern, auf denen nur Personen dargestellt sind, ist es schwierig inhaltliche Details bestimmten. Des Weiteren gelingt Identifizierung von genauen Inhalten eher für aktuelle Themen. Für den gezielten Nachrichtenkonsum wird die Kombination von Bild und Text bevorzugt. Der „Nur-Bilder-Modus“ funktioniert nur, wenn der Leser genügend Zeit und Interesse mitbringt.

## **6. Fazit**

## Literatur

[Bleske 2012] BLESKE, Christian: JavaScript vs. C#. In: *Mobile Developer* 7 (2012), S. 16–20

[Microsoft 2013a] MICROSOFT: *Navigationsdesign für Windows Store-Apps*. <http://msdn.microsoft.com/de-de/library/windows/apps/hh761500.aspx>. Version: 2013. – zuletzt geprüft: 17.08.2013

[Microsoft 2013b] MICROSOFT: *Quickstart: Using single-page navigation (Windows Store apps using JavaScript and HTML)*. <http://msdn.microsoft.com/en-us/library/windows/apps/hh452768.aspx>. Version: Juni 2013. – zuletzt geprüft: 20.08.2013

[Microsoft 2013c] MICROSOFT: *Windows RT: Häufig gestellte Fragen*. <http://windows.microsoft.com/de-de/windows/windows-rt-faq>. Version: 2013. – zuletzt geprüft: 31.08.2013

[O'Brian 2013] O'BRIAN, Tim: *Web, native, and déjà vu*. <http://channel9.msdn.com/Blogs/Vector/Web-native-and-dj-vu>. Version: Januar 2013. – zuletzt geprüft: 29.07.2013

[Pachal 2012] PACHAL, Pete: *The Philosophy Behind Windows 8, From One of Its Creators*. <http://mashable.com/2012/10/25/philosophy-windows-8/>. Version: Oktober 2012. – zuletzt geprüft: 31.08.2013

[WinBeta 2013] WINBETA: *Windows Phone has 64% of the top 100 popular apps on Apple's iOS platform*. <http://www.winbeta.org/news/windows-phone-has-64-top-100-popular-apps-apples-ios-platform>. Version: August 2013. – zuletzt geprüft: 31.08.2013

# A. Detaillierte Evaluierung Bild-Inhalt Relation

## Politik Artikel

- 1 Deutsche Politik einheitlich gegen Militärschlag in Syrien
 

grob	100%
genau	0%
- 2 Wir werden ein schlechtes Erbe antreten
 

grob	80%
genau	80%
- 3 Deutschland will von UN-Ausspähung nichts gewusst haben
 

grob	20%
genau	20%
- 4 Islamisten wollen Giftgasangriff rächen
 

grob	80%
genau	40%
- 5 Bo Xilai nennt Zeugen Lügner
 

grob	100%
genau	80%
- 6 Prozess gegen Anführer der Muslimbrüder offenbar vertagt
 

grob	80%
genau	0%

## Wissen Artikel

- 1 Skelett aus Eiweiß-Würmchen hält Chromosomen zusammen
 

grob	60%
genau	20%
- 2 Graf Zeppelin hat das Zeppelin nicht erfunden
 

grob	100%
genau	0%
- 3 Deutsche Fliegennetze für Afrikas Rinder
 

grob	60%
genau	0%
- 4 Waldbrand bedroht Stromversorgung
 

grob	100%
genau	100%
- 5 Hat nur der Mensch Sex in der Missionarsstellung?
 

grob	80%
genau	0%
- 6 Fukushima ersäuft in verstrahltem Wasser
 

grob	80%
genau	80%

P1	Artikel					
	1	2	3	4	5	6
Politik grob	ja	ja	nein	ja	ja	ja
Politik genauer	nein	ja	nein	ja	ja	nein
Wissen grob	nein	ja	ja	ja	nein	ja
Wissen genauer	nein	nein	nein	ja	nein	ja

## A. Detaillierte Evaluierung Bild-Inhalt Relation

P2	Artikel					
	1	2	3	4	5	6
Politik grob	ja	nein	nein	ja	ja	ja
Politik genauer	nein	nein	nein	nein	nein	nein
Wissen grob	ja	ja	ja	ja	ja	nein
Wissen genauer	nein	nein	nein	ja	nein	nein

P3	Artikel					
	1	2	3	4	5	6
Politik grob	ja	ja	ja	ja	ja	ja
Politik genauer	nein	ja	ja	nein	ja	nein
Wissen grob	nein	ja	ja	ja	ja	ja
Wissen genauer	nein	nein	nein	ja	nein	ja

P4	Artikel					
	1	2	3	4	5	6
Politik grob	ja	ja	nein	ja	ja	nein
Politik genauer	nein	ja	nein	ja	ja	nein
Wissen grob	ja	ja	nein	ja	ja	ja
Wissen genauer	ja	nein	nein	ja	nein	ja

P5	Artikel					
	1	2	3	4	5	6
Politik grob	ja	ja	nein	nein	ja	ja
Politik genauer	nein	ja	nein	nein	ja	nein
Wissen grob	ja	ja	nein	ja	ja	ja
Wissen genauer	nein	nein	nein	ja	nein	ja

Auswertung	ges. Anzah	richtig	%
Gesamt grob	60	47	78,3
Gesamt genauer	60	21	35
P1 /grob	12	9	75
P1 /genau	12	5	41,7
P2/grob	12	9	75
P2/genau	12	1	8,33
P3/grob	12	11	91,7
P3/genau	12	5	41,7
P4/grob	12	9	75
P4/genau	12	6	50
P5/grob	12	9	75
P5/genau	12	4	33,3

*B. CD Inhalt*

**B. CD Inhalt**

# Eigenständigkeitserklärung

Ich versichere, die vorliegende Arbeit selbständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht.

*Hamburg, 03.09.2013*

*Malte Modrow*