



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

1. Einleitung	1
2. Windows 8	2
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	7
3. Konzeption der App	8
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	12
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	16
3.4.2. Windows 8 App mit Webtechnologien	17
4. Umsetzung der App	18
4.1. Rastervorlage	18
4.1.1. HTML Dateien	18
4.1.2. Javascript Dateien	19
4.1.3. CSS Dateien	19
4.2. Datenanbindung	19
4.2.1. ZEIT ONLINE Datenbestand	19
4.2.2. Daten verfügbar machen	21
4.3. Darstellung der Daten	24
4.3.1. Die Hubansicht	24
4.3.2. Die Ressortansicht	26
4.3.3. Die Artikelansicht	26
4.4. Features	28
4.4.1. Artikeltitel ausblenden	28

4.4.2. Schriftgröße verändern	30
4.4.3. Semantic Zoom	30
4.5. Entwicklungschronologie	32
5. Evaluierung	34
5.1. Fokusgruppe	34
5.2. Ablauf	34
5.3. Ergebnisse	36
5.3.1. Intuitivität der App	36
5.3.2. Der nur Bilder Modus	37
6. Fazit	39
Literatur	40
A. Detaillierte Evaluierung Bild-Inhalt Relation	i
B. CD Inhalt	iii

Abbildungsverzeichnis

1.	Der Startbildschirm von Windows 8	2
2.	Die Geste zum Aufrufen der Charms-Menüleiste.	3
3.	Die Gesten zum Durch-wechseln der Apps.	4
4.	Geste zum Öffnen der unteren und oberen Menüleisten.	4
5.	Ein möglicher Startscreen von Windows Phone 8.	6
6.	Verfügbarkeit der Top 100 iOS Apps auf anderen Systemen (WinBeta, 2013).	7
7.	Die Homepage von ZEIT ONLINE.	8
8.	Die mobile Webseite von ZEIT ONLINE.	9
9.	Die iPad App zeigt die aufbereiteten Artikel der gedruckten Ausgabe der ZEIT.	10
10.	Die Webapp zeigt genau wie die iPad App, die aufbereiteten Artikel der gedruckten ZEIT.	11
11.	Die iPhone App zeigt sowohl Inhalte von ZEIT ONLINE, als auch von der ZEIT.	11
12.	Flaches Navigationssystem (Microsoft, 2013a).	13
13.	Hierarchisches Navigationssystem (Microsoft, 2013a).	14
14.	Schema des „Semantic Zoom“ (Microsoft, 2013a).	15
15.	Die Projektstruktur in Visual Studio.	18
16.	Die Hubansicht.	24
17.	Die Einzelressortübersicht (hier das Reisen Ressort).	26
18.	Die Artikelansicht.	27
19.	Untere Menüleiste mit der Funktion zum Artikeltitel ausschalten.	29
20.	Das Reisen Ressort ohne Artikeltitel.	29
21.	links: Schriftgröße Button im unteren Menü mitte: Das Flyout-Menü mit allen Schaltflächen. rechts: Das Flyout-Menü beim Aufruf wenn schon die größte Größe eingestellt ist.	30
22.	Die semantic Zoom Ansicht.	31
23.	Die Testbilder aus dem Politik- und Wissen-Ressort.	35

Tabellenverzeichnis

1.	Übersicht der Testpersonen.	34
----	-------------------------------------	----

Listings

1.	Das XML eines Teaserelements	20
2.	Initialisierung der Binding-List.	21
3.	Auszug aus dem Ressorts Array.	21
4.	Parsen und speichern der Daten.	23
5.	Die wichtigsten Markupelemente der Hubansicht.	25
6.	Template bei der Listview registrieren.	25
7.	Setzen der Datenquelle und des Layouts.	26
8.	Das Einfügen von Titel TeaserText und Bild.	27
9.	Vermeidung von Infoboxen Twitternachrichten und Hyperlinks in den Artikeln.	28
10.	Der Semantic Zoom Wrapper.	31

Abkürzungsverzeichnis

XAML	Extensible Application Markup Language	17
CMS	Content Management System	19
XML	Extensible Markup Language	19
XSLT	Extensible Stylesheet Language Transformations	19
HTML	Hypertext Markup Language	17
XHR	XMLHttpRequest	21
WinJS	Windows Library for JavaScript	17
GUI	Graphical User Interface	12

1. Einleitung

1. Einleitung

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-Computer¹ 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 eine neues, kontextsensitives Ribbon-Menü². Neu ist hingegen das "Modern User Interface"(Modern UI, früher Metro), eine für Touch-Gesten optimierte Oberfläche. Obwohl die neue Oberfläche für Touch-Gesten optimiert ist, kann sie ebenso 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. Viele bunte, viereckige Kacheln, in verschiedenen Größen. So sieht die neue Startoberfläche von Windows 8 aus (siehe Abbildung 1). Auf dem Startbildschirm sind die verschiedenen Programme angeordnet.



Abbildung 1: Der Startbildschirm von Windows 8.

Dies können „normale“ Programme sein, die auf dem Desktop laufen oder es können 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,

¹Tablet-Computer: kleiner, flacher, tragbarer Computer mit einem Touchscreen. Es wird von nun an, der Einfachheit halber die englische Version „Tablet“ benutzt.

²Auch bekannt als Menüband oder Multifunktionsleiste.

2. Windows 8

wenn auch ein sehr spezielles, Programm (App), welches in der neuen Oberfläche gestartet und ausgeführt werden kann.

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, „zur Startansicht“, 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.

Um bequem zwischen den geöffneten Apps zu wechseln, können zwei verschiedene Gesten benutzt werden. Abbildung 3 zeigt die beiden Möglichkeiten. Um eine Liste wie sie die Abbildung auf der linken Seite zeigt 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 runter bzw. hoch 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. Es ist egal 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 Einstellun-

2. Windows 8

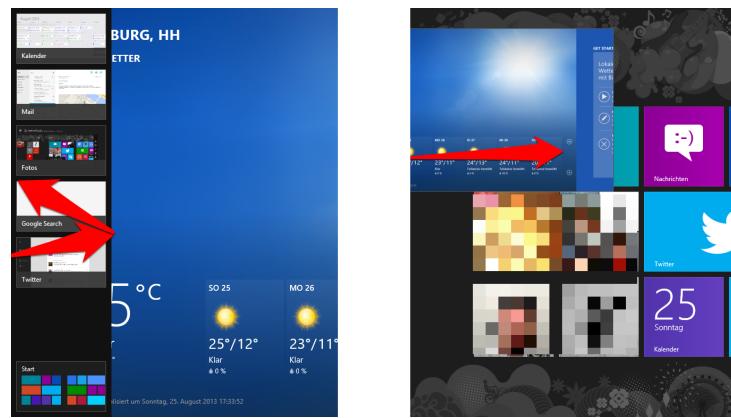


Abbildung 3: Die Gesten zum Durch-wechseln der Apps.

gen für die aktuelle Ansicht, in welcher sich die App gerade befindet. In der oberen Leiste ist typischer Weise eine Art von Navigation untergebracht (falls es eine obere Leiste gibt). Abbildung 4 zeigt die obere und untere Menüleisten am Beispiel der vorinstallierten Wetter App.



Abbildung 4: Geste zum Öffnen der unteren und oberen Menüleisten.

2.2. Die Windows 8 Philosophie

Windows 8 ist wohl der radikalste Umbruch bei Microsoft 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

2. Windows 8

zeigen stets aktuelle Informationen ohne dass der User den Browser öffnen muss (Pachal, 2012).

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 Moreau³

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, welche mit einer x86/x64 Architektur ausgestattet sind. Auf den ersten Blick sieht Windows RT, von der Bedienoberfläche her, genau aus wie Windows 8. Der größte und signifikanteste Unterschied zu Windows 8 ist jedoch, dass aufgrund der ARM-Architektur auf Windows RT nur Programme und Apps aus dem Windows-Store installiert und ausgeführt werden können. Das bedeutet, 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 verfügbar ist, allerdings laufen hier nur die vorinstallierten Office Produkte von Microsoft, sowie der Microsoft Browser *Internet Explorer*. Außerdem können nicht alle Peripheriegeräte wie z.B. Mäuse oder Tastaturen mit Windows RT verwendet. Nur speziell für Windows RT zertifizierte Geräte können benutzt werden. (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

³Director of design and research for Windows

2. Windows 8

muss vorher genau bedacht werden ob das gewählte Ökosystem den eigenen Anforderungen entspricht. Fragen die sich 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 sehen die Dienste zum Speichern von Daten in der Cloud aus? Es fällt auf, dass heute nicht mehr nur ein Gerät gekauft wird, sondern das an diesem Gerät oder Geräten meist ein ganzes Ökosystem mit geliefert wird, egal 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, sowie den E-Mail Dienst *outlook.com*. Viele dieser Dienste existierten schon vor der Einführung von Windows 8. Aber erst mit dieser Einführung wurden die Dienste und Betriebssysteme einer einheitlichen „Designkur“ unterzogen und enger auf einander abgestimmt. Außerdem benötigt man, im Gegensatz zu vorher, nur noch ein Benutzerkonto 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 auch keine Windows 8 Apps unter Windows Phone. Auch beim Entwickeln der Apps kann nicht gleichzeitig für beide Plattformen entwickelt werden. Abbildung 5 zeigt einen Startbildschirm von Windows Phone 8

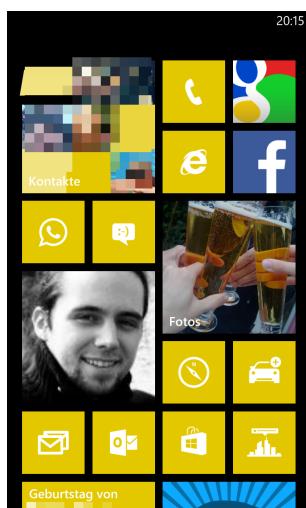


Abbildung 5: Ein möglicher Startscreen von Windows Phone 8.

2. Windows 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 ist es der *App Store* und , 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 für, bzw. gegen ein bestimmtes Gerät aus der Windows-Welt, mit der Begründung es gäbe zu wenig Apps im Windows Store.

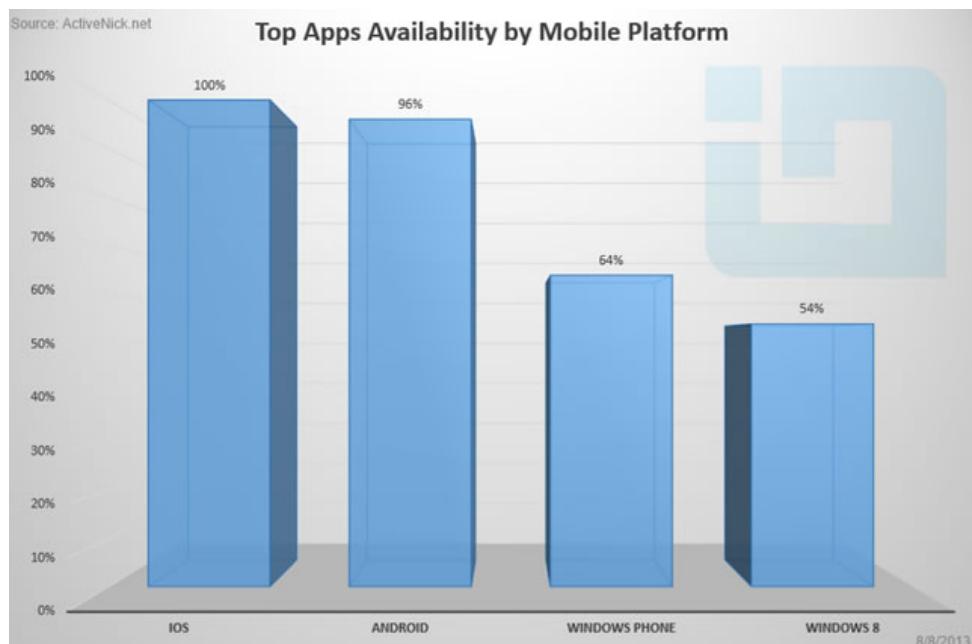


Abbildung 6: Verfügbarkeit der Top 100 iOS Apps auf anderen Systemen (WinBeta, 2013).

Abbildung 6 zeigt die Verfügbarkeit der Top 100 iOS Apps⁴ 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 App wie z.B. *Instagram*, *Pinterest*, *Instapaper* oder *Readability* schnellstmöglich nachzuliefern. Microsoft bietet den Entwicklern teilweise sogar Geld damit sie eine offizielle „große“ App entwickeln (WinBeta, 2013).

⁴Liste: <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

3. Konzeption der App

In dieser Sektion wird zunächst ein Blick auf die ZEIT ONLINE Webseite und auf bereits vorhandene ZEIT ONLINE Apps geworfen. Anschließend 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 Möglichkeit ausfallen. Dazu müssen zunächst die Ziele der App ausgeführt werden. Anschließend muss sich entschieden werden mit welcher Technologie bzw. Programmiersprache gearbeitet werden soll. Zuletzt wird noch ein detaillierterer Blick auf das Herzstück der App geworfen. Das Navigationskonzept. Hierbei ist eine der wichtigsten Fragen: „Wie kann ich auch bei ggf. großen Datensätzen eine übersichtliche, intuitive Struktur schaffen die sich in das Gesamtkonzept von Windows 8 eingliedert?“

3.1. ZEIT ONLINE

ZEIT ONLINE ist das Internetangebot der Wochenzeitung DIE ZEIT. Berichtet wird über tagesaktuelle Geschehnisse, von einer eigenen Redaktion, welche größtenteils in Berlin sitzt. Den Themenschwerpunkt bildet das Politikressort. Um die ZEIT ONLINE Inhalte, sowie auch die Inhalte der gedruckten ZEIT möglichst vielen Nutzern zugänglich zu machen, betreibt (bzw. lässt betreiben) ZEIT ONLINE verschiedene Webseiten und Apps. Im folgenden soll ein kurzer 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, welche laufend aktualisiert werden.



Abbildung 7: Die Homepage von ZEIT ONLINE.

3. Konzeption der App

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 sehen 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.

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

Da zunehmend immer mehr Leser sie 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 Teaserelemente 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 recht, die Startseite, die Artikelansicht und das Navigationsmenü für die Ressorts dargestellt.

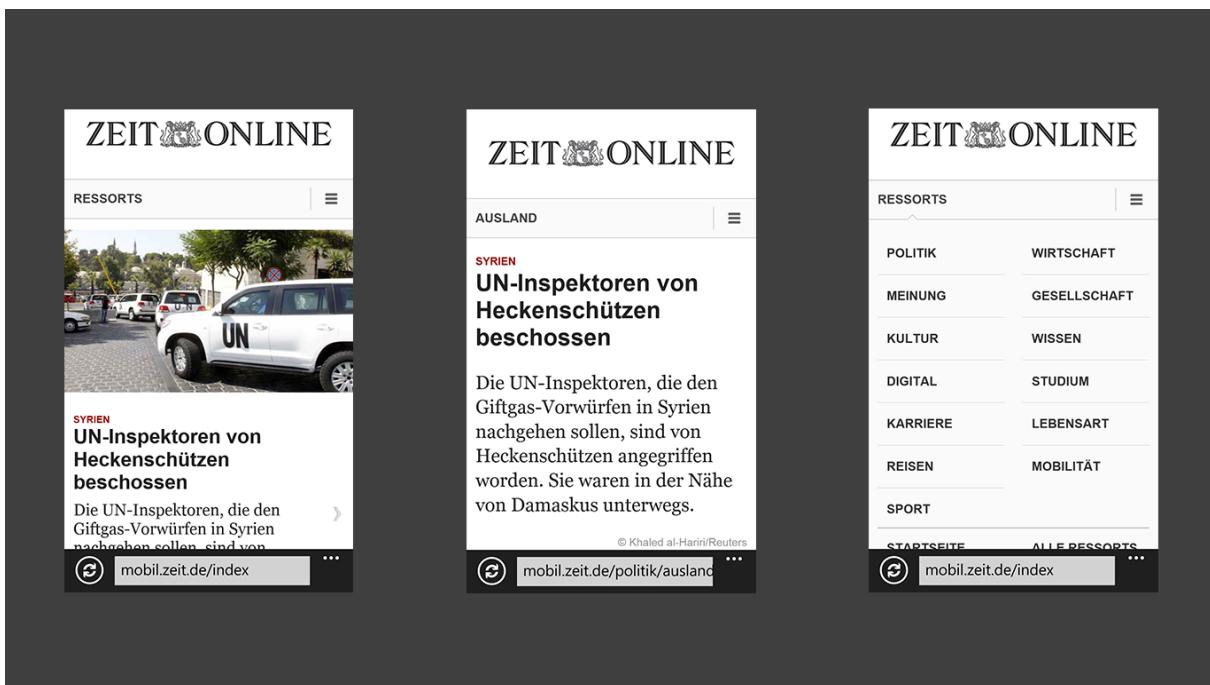


Abbildung 8: Die mobile Webseite von ZEIT ONLINE.

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, sowie das ZEIT Magazin an. Die Ausgaben können sowohl im Abo, 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.

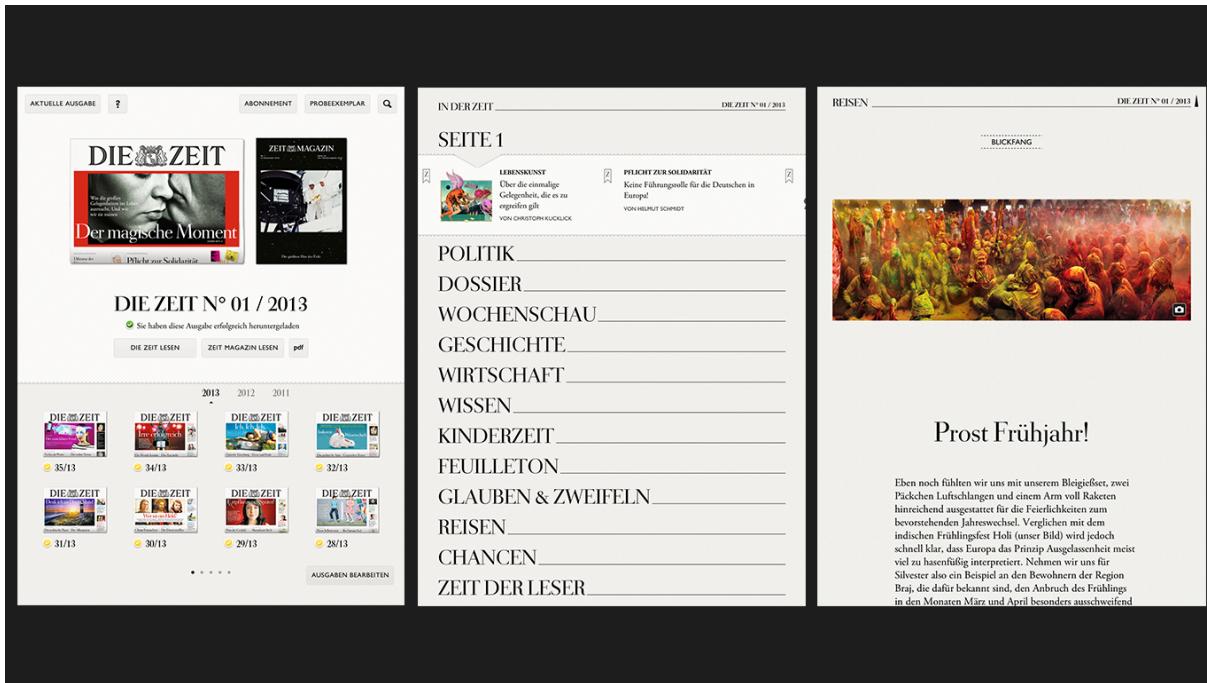


Abbildung 9: Die iPad App zeigt die aufbereiteten Artikel der gedruckten Ausgabe der ZEIT.

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 erreichbar. Des weiteren gibt es einen Android- und einen Kindle-Wrapper, sodass die App auch über den *Google Play Store*, sowie den *Amazon Store* zu beziehen ist.

3. Konzeption der App

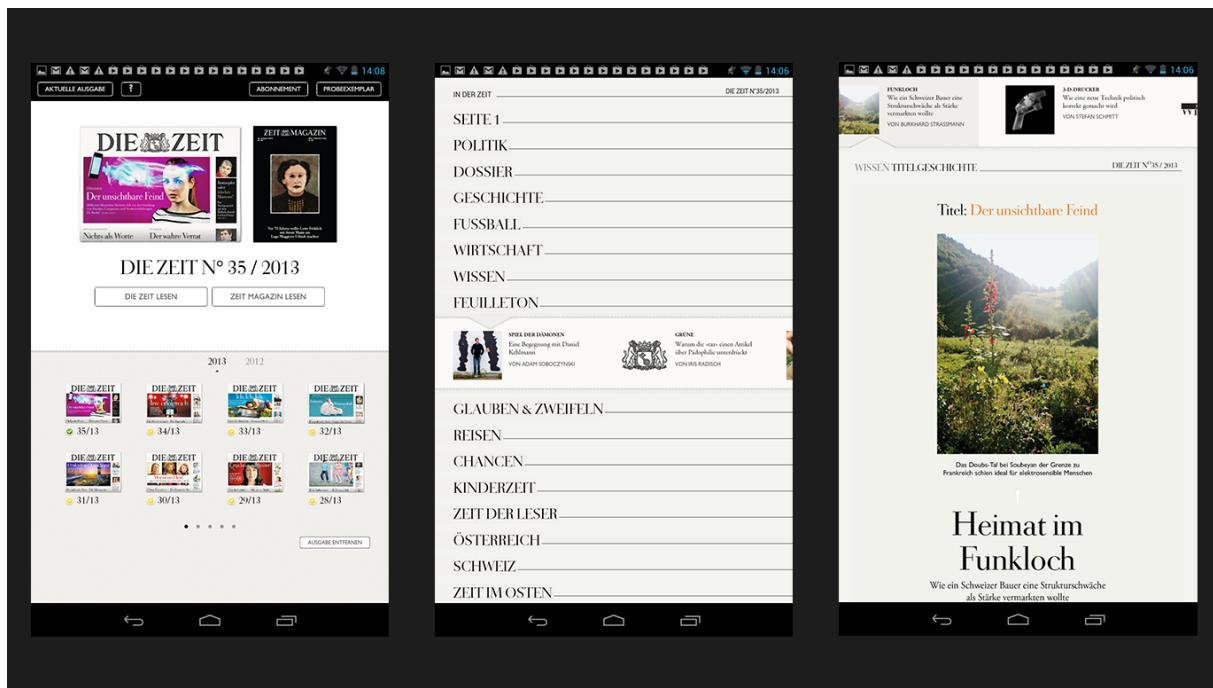


Abbildung 10: Die Webapp zeigt genau wie die iPad App, die aufbereiteten Artikel der gedruckten ZEIT.

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

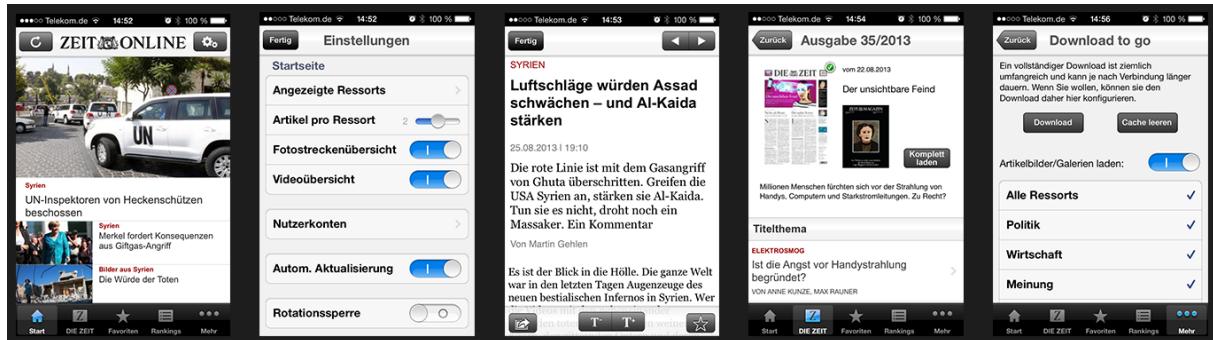


Abbildung 11: Die iPhone App zeigt sowohl Inhalte von ZEIT ONLINE, als auch von der ZEIT.

3. Konzeption der App

Abbildung 11 zeigt eine Bildstrecke der iPhone App. Das zweite Bild von links zeigt einige der möglichen Einstellungen, welche 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 Ziel der zu erstellenden App ist, den Inhalt der ZEIT ONLINE Website auf ansprechende Art und Weise in einer Windows 8 Applikation darzustellen. Das Layout und die Darstellung der Inhalte soll 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. Das heißt andere Inhalte wie z.B. Bildergalerien, Infografiken, Blogartikel oder Quizze werden von der App nicht erfasst und nicht dargestellt. Hintergrund ist, die App möglichst einfach zu halten da es in der Fragestellung um die Relation zwischen den Aufmacherbildern und dem dahinter liegenden Artikeltext 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.

Der User soll die Möglichkeit haben die standardmäßig vorhandenen Artikeltitel auszublenden um so, wenn gewünscht, einen rein visuellen Eindruck der Artikelbilder zu bekommen. Hierzu soll einen Schalter in der Menüleiste am unteren Bildrand geben. Wenn der User sich im Artikel befindet soll er die Möglichkeit haben die Schriftgröße in gewissen Maß selbst zu bestimmen, da die App eventuell auf Monitoren mit verschiedenen großen Auflösungen ausgeführt werden wird oder die Sehkraft des Users nicht mehr ausreicht um eine normal große Schrift zu erkennen und zu lesen. So wird in den Artikeln ein gewisses Maß an Barrierefreiheit gewährleistet.

Das übergeordnete Ziel der App ist es, eine rudimentäre Nachrichten Applikation für Windows 8 zu erstellen. Gleichzeitig soll eine Umgebung geschaffen werden, die es erlaubt Untersuchungen anzustellen, in wie weit es möglich ist allein durch das Betrachten der Aufmacherbilder auf den Inhalt der jeweiligen Artikel zu schließen (siehe Sektion 5.3.2 auf Seite 37). Außerdem soll die App dazu dienen, zu erforschen, wie eine Nachrichten Applikation in der Modern UI Oberfläche von Windows 8 erstellt wird und welche design- als auch funktionstechnischen Vorgaben von Microsofts vorhanden sind. Sprich, wie eine App mit Nachrichteninhalten nach Microsofts Sichtweise auszusehen hat.

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

3. Konzeption der App

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 der 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.

3.3.1. Flaches Navigationssystem

Das flache Navigationssystem wird in vielen Windows Store Apps verwendet, häufig in Spielen, Browsern 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 wenn ein schneller Wechsel zwischen wenigen Seiten oder Registerkarten möglich sein soll (Microsoft, 2013a).

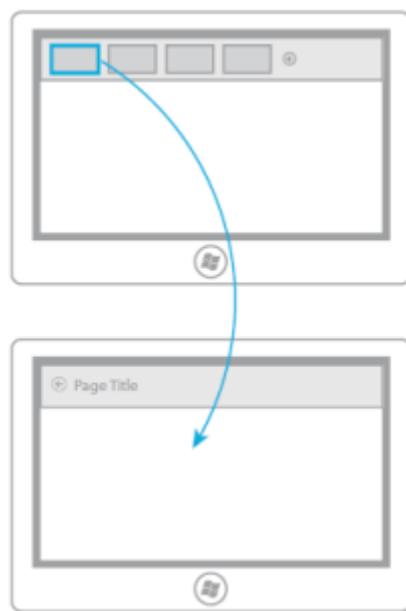


Abbildung 12: Flaches Navigationssystem (Microsoft, 2013a).

Abbildung 12 zeigt wie das flache Navigationssystem funktioniert. Am oberen Rand befindet sich eine, nicht immer sichtbare Navigationsleiste mit den verschiedenen Registerkarten oder Seiten. Die Leiste wird angezeigt, wenn der User vom unteren oder oberen Bildrand streift (siehe Sektion 2.1). Wenn der User die Seite wechselt möchte geschieht dies direkt über die Navigationsleiste, d.h. es gibt meist keinen permanenten Zurück-Button.

Ein flaches Navigationssystem eignet sich nicht unbedingt für eine Nachrichten App wie die, die in dieser Arbeit konzipiert und umgesetzt werden soll, weil es zu viele

3. Konzeption der App

Bereiche (Ressorts) gibt, welche sich zudem sehr ähneln. Hier ist es angebracht ein hierarchisches Navigationssystem zu verwenden.

3.3.2. Hierarchisches Navigationssystem

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

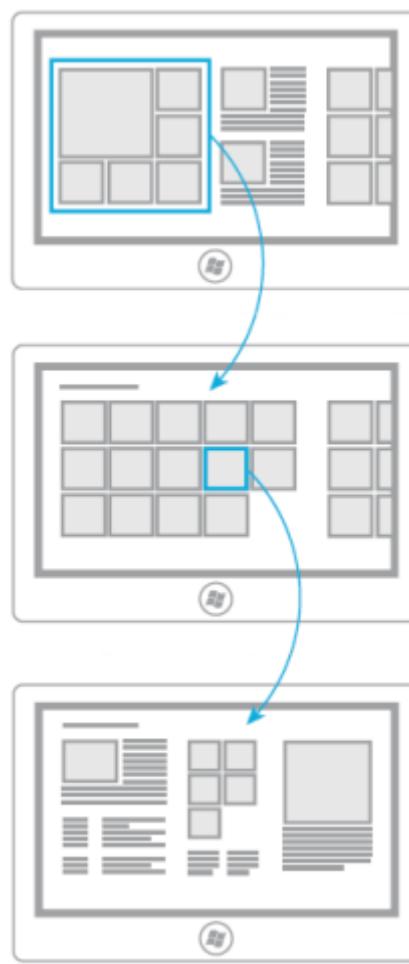


Abbildung 13: Hierarchisches Navigationssystem (Microsoft, 2013a).

Das Schema eines hierarchischen Systems ist in Abbildung 13 dargestellt. Der Einstiegspunkt in die App ist die sogenannte „Hubseite“ hier ganz oben in der Abbildung zu sehen. Auf der Hubebene werden 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 jeden Bereich gleich sein. Es kann das Nutzungserlebnis sogar verbessern und vielfältiger machen wenn unterschiedliche Darstellungen (z.B. in Höhe, Breite oder Anzahl der Objekte) für die Bereiche genutzt

3. Konzeption der App

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.

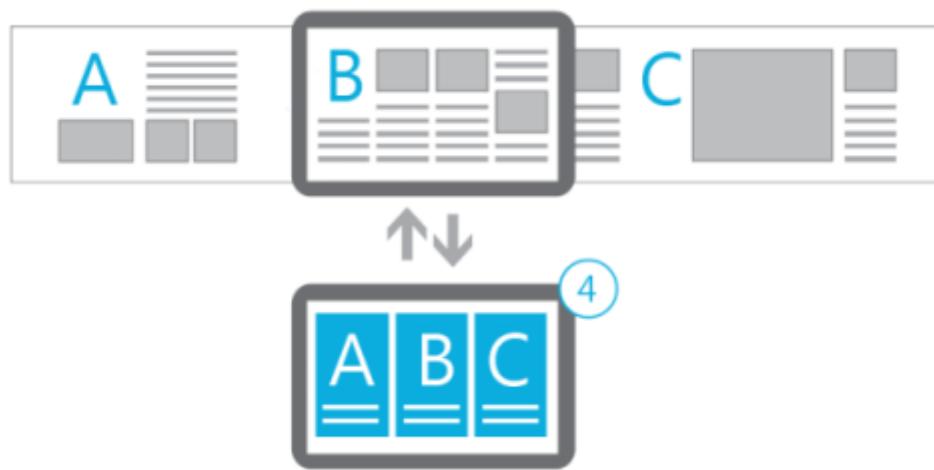


Abbildung 14: Schema des „Semantic Zoom“ (Microsoft, 2013a).

Befinden sich in der Hubansicht, trotz der Zusammenfassung der Bereiche noch zu viele Bereiche und es dauert zu lange bis zum Ende 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 mit der „Pinch to Zoom (out)⁵“ 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 überall aus der Hubansicht aufgerufen werden.

⁵Zwei Finger (Berührungs punkte), die sich auf dem Bildschirm auf einander zu bewegen.

3.3.3. Menü und Features

Da die App einen explorativen Charakter hat wird sich bei den Funktionen und Features auf das nötigste beschränkt. Es soll eine untere App-Leiste geben, in welcher, je nach dem in welcher Ansicht sich der User gerade befindet verschiedene Optionen angezeigt werden. Diese Leiste ist nicht dauerhaft zu sehen und kann, wenn man mit der Maus arbeitet durch einen Rechtsklick aufgerufen werden. 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 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, sowie die Schriftgröße wieder auf ihren Startwert zu setzen.

3.4. Umsetzungsmöglichkeiten in Visual Studio

Vor dem Entwickeln einer Windows 8 App muss sich 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. Das heißt 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 Windows 8 selbst und zum anderen wird die neueste Version von Visual Studio, Visual Studio 11⁶, benötigt. Visual Studio steht in der Express Version für Windows 8 kostenlos zur Verfügung. Des weiteren muss sich 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.

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 sinnige Definition liegt irgendwo zwischen diesen beiden Varianten. Eine App ist dann nativ, wenn sie Geräte-, Betriebssystem- oder Laufzeitumgebungsabhängig ist. Das heißt, sie ist für ein spezielles Gerät entwickelt und kann nur auf diesem ausgeführt werden. Sie kann dabei alle Geräte- oder Betriebssystemspezifischen Funktionen nutzen, es ist jedoch egal wie nah an der Hardware tatsächlich programmiert wurde

⁶Visual Studio 2011 war die aktuellste Version beim Erstellen dieser Arbeit.

3. Konzeption der App

(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 werden oder man lässt es sich automatisch generieren. 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, sowie 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 benutzen. Das bedeutet, Apps können mit Hypertext Markup Language (HTML), CSS und Javascript erstellt werden. Das Markup wird mit HTML erstellt. Dieses mit CSS gestylt und die Programmlogik wird 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 extra 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. Es findet so, in gewisser Weise, eine Zusammenführung von .NET- und Webentwicklern statt. Ganz ohne Lernaufwand kommen allerdings auch die Webentwickler nicht aus. Es muss sich trotzdem noch in die neue API-Technologie eingearbeitet werden (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 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

4.1. Rastervorlage⁷

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.

Es wird von Microsoft empfohlen, dass Windows-Store Apps, welche mit HTML und Javascript erstellt werden das single-page Navigationmodell verwenden. Das bedeutet, die Navigation erfolgt nicht über Hyperlinks, sondern die verschiedenen Inhalte werden direkt in die Seite nachgeladen (ähnlich wie ein iframe). Auf diese Weise gibt keine sichtbare Unterbrechung beim Navigieren (der Bildschirm wird nicht kurz weiß beim Navigieren). Außerdem erzielt man auf diese Weise eine bessere Performance und die App fühlt sich mehr „app-like“ an. Die Rastervorlage verwendet das single-page Navigationsmodell (Microsoft, 2013b). Abbildung 15 zeigt die Dateistruktur des Projektes. Anschließend wird kurz erläutert wofür die einzelnen Dateien da sind.

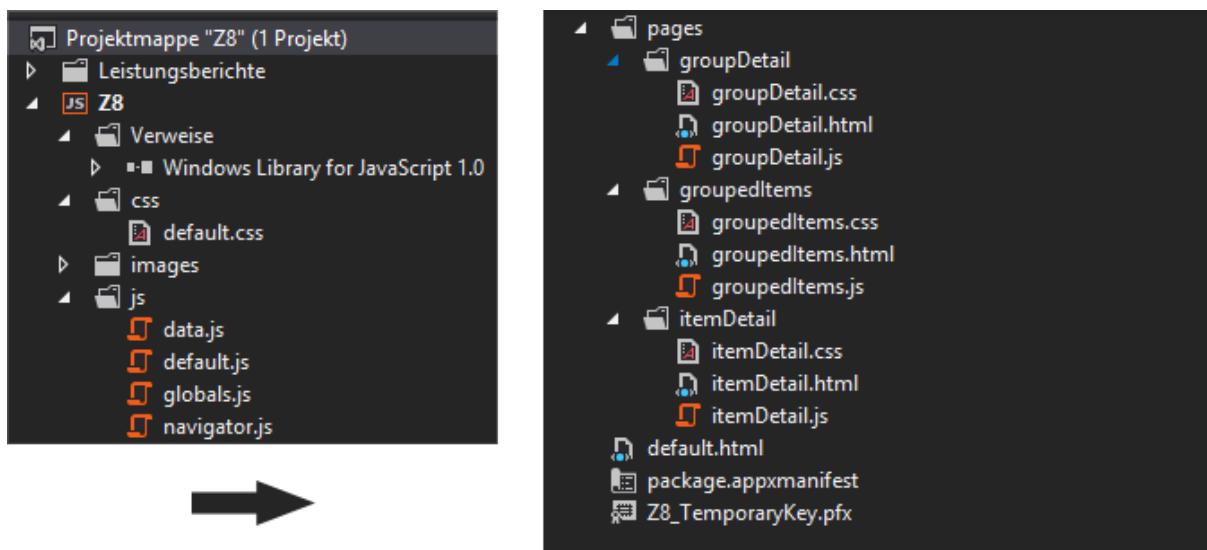


Abbildung 15: Die Projektstruktur in Visual Studio.

4.1.1. HTML Dateien

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

default.html, 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

⁷Die Namen der verwendeten Dateien in der Rastervorlage wurden mittlerweile von Microsoft geändert.

4. Umsetzung der App

herein geladen werden).

groupedItems.html , ist der Einstiegspunkt in die App (die Hubansicht).

groupDetail.html , zeigt alle Elemente eines Bereichs.

itemDetail.html , Einzelansicht eines Elements.

4.1.2. Javascript Dateien

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

default.js , legt fest wie sich die App beim Start verhält.

groupedItems.js | groupDetail.js | itemDetail.js , sind die Javascript Dateien, welche das Verhalten für die gleichnamigen HTML Dateien festlegen.

navigator.js , implementiert das hierarchische Navigationssystem, sowie das single-page Navigationmodell.

data.js , stellt die benötigten Daten für den Rest der App bereit.

4.1.3. CSS Dateien

default.css , enthält Styles für Inhaltshostseite, sowie für die gesamte App.

groupedItems.css | groupDetail.css | itemDetail.css ,

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

4.2. Datenanbindung

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. Dies ist ein grober Überblick wie sich die Webseite von ZEIT ONLINE zusammensetzt.

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

4. Umsetzung der App

jeweiligen Ressorts (z.B. die Hauptseite des Politikressorts), 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, welcher kurz 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 (Inhalt) in Form von Paragraphen-Tags (`<p>`) enthalten. Dies ist ein konkreter Einblick wie die rohen Daten aussehen, welche für die App verwendet werden. 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 Daten Einlesen und Verarbeiten geschieht in der „data.js“ Datei. Es werden alle Teaser Elemente 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: "tbd",
6     acquireSyndication: acquireSyndication, dataPromise: null
7   }
8 ]
```

Listing 3: Auszug aus dem Ressorts Array.

Anschließend wird eine weitere Funktion von WinJS verwendet, die WinJS.Promises. 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

4. Umsetzung der App

alle Promises vorhanden sind (alle URL waren erreichbar), wird die Datenverarbeitung fortgesetzt.

Mit den vorhandenen und validen XHR Responses können anschließend alle Teaserlemente 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 erstellte WinJS.Binding.List geschrieben (Zeilen 24-28). Die dargestellte Funktion ist nicht vollständig und wurde aus Übersichts- 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
8          //only articles with an image are allowed
9          if (teaser.getAttribute("contenttype") == "article" && teaser.
10             querySelector("image") != null) {
11              // Get the title
12              var teaserTitle = teaser.querySelector("title").textContent;
13
14              // Process the content so that it displays nicely.
15              var staticContent = toStaticHTML(teaser.querySelector("description
16                  ").textContent);
17
18              //Get and build the image path
19              var teaserImageEl = teaser.querySelector("image");
20              var imagebasePath = teaserImageEl.getAttribute("base-id");
21              var splitImagePath = imagebasePath.split("/");
22              var imageNameSmall = splitImagePath[splitImagePath.length - 2] + "
23                  -220x124.jpg";
24              var imageNameBig = splitImagePath[splitImagePath.length - 2] + "
25                  -540x304.jpg";
26              var imagePathSmall = imagePath + imageNameSmall;
27              var imagePathBig = imagePath + imageNameBig;
28
29              // Store the teaser element info we care about in the array.
30              teaserElements.push({
31                  group: ressort, key: ressort.title, title: teaserTitle,
32                  backgroundImage: imagePathBig, teaserText: staticContent
33              });
34      }
35  }

```

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 wenn der User auf einen Artikel klickt.

4.3. Darstellung der Daten

4.3.1. Die Hubansicht

Wenn die App startet befindet sich der User in der Hubansicht. Hier werden ihm für alle Hauptressorts die ersten sechs Artikel angezeigt, in der Reihenfolge wie sie auch auf der Webseite zu finden sind (siehe Abbildung 16).

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 welcher das Verhalten (die Logik) der Hubseite programmiert wird.

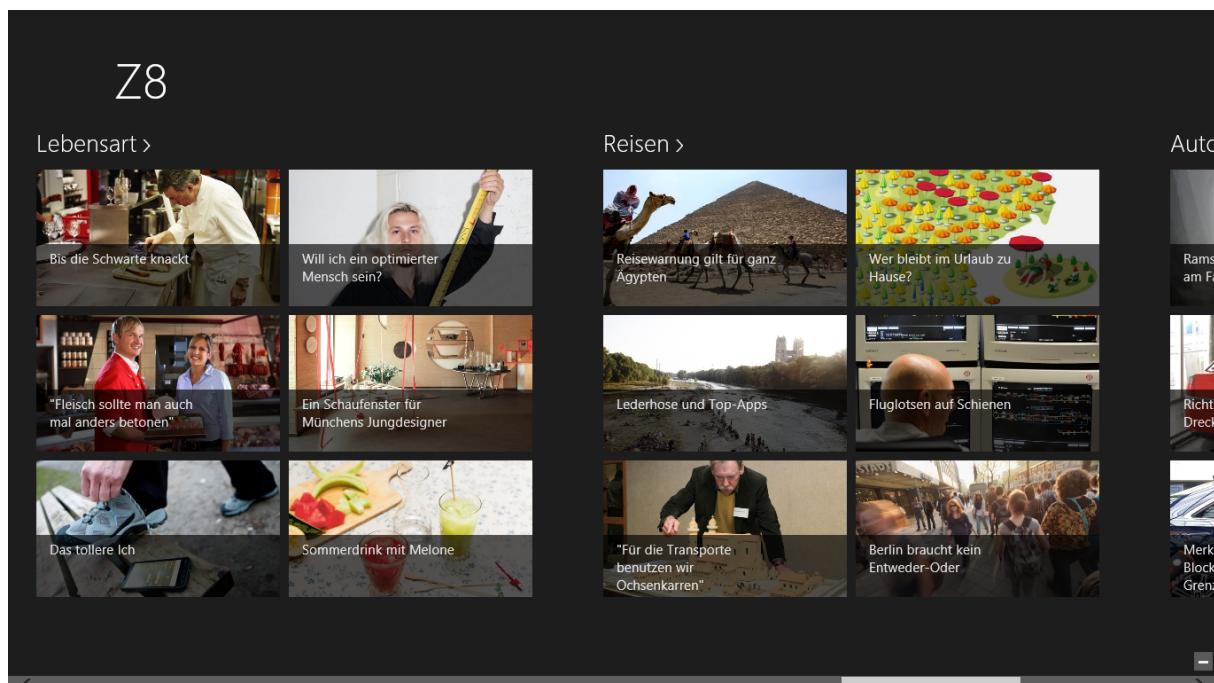


Abbildung 16: Die Hubansicht.

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. Dies bedeutet, es akzeptiert Daten aus einer *WinJS.Binding.List*, wie sie in der Rohdatenverarbeitung genutzt wurde. Im unteren Teil ab Zeile 10 ist 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="{
13       selectionMode: none}"></div>
14 </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 die *WinJS.ListView* zu bekommen 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 16 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 17). Der User gelangt hierhin wenn er in der Hubansicht auf den Ressortnamen im Headerbereich klickt.

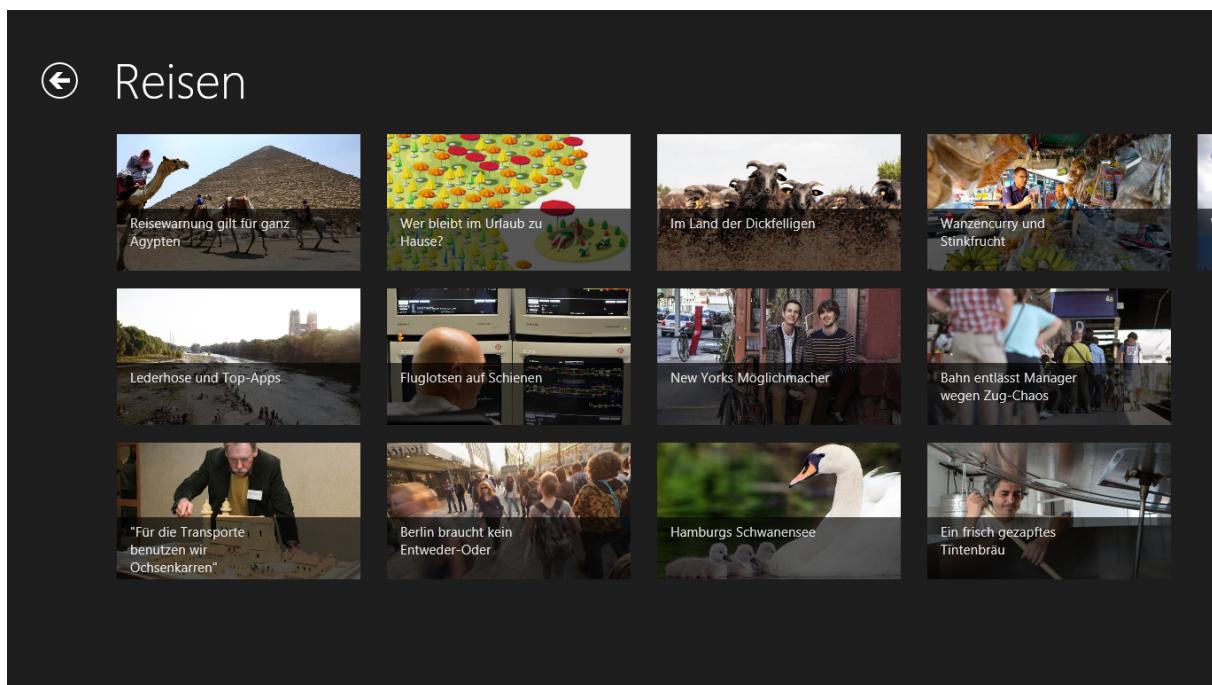


Abbildung 17: Die Einzelressortübersicht (hier das Reisen Ressort).

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 ist, 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 18). Dargestellt wird das Ressort, der Titel, der TeaserText, das Aufmacherbild in groß, die Bildunterschrift, das Copyright für das Bild, der Artikeltext,

4. Umsetzung der App

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 gefüllt. In der *ready()* Funktion der *itemDetail.js* Datei werden zunächst die bereits vorhandenen Informationen in das HTML eingefügt.



Abbildung 18: 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, welche in jedem Teaser-Element gespeichert ist. Auf diese Weise bekommt man das XML der Artikel, wie es auch von der Webseite verwendet wird. Beim Parsen des Artikel-XMLs müssen drei Dinge besonders beachtet werden.

```
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 Teaser-Text und Bild.

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. Ebenso ist es möglich, dass Nachrichten vom Mikroblogging Dienst Twitter eingebunden werden. Diese Abschnitte arbeiten ebenfalls mit Paragraphen-Tags. Da Infoboxen und Twitternachrichten sich inhaltlich nicht direkt in den Artikeltext eingliedern, müssen diese Abschnitte vorher herausgefiltert werden. Des Weiteren sind

in den meisten Artikeln auf der ZEIT ONLINE Webseite Hyperlinks, entweder auf andere Artikel oder auch andere Webseiten, vorhanden. Diese sind genau wie Infoboxen und Twitternachrichten unerwünscht, da sie, wenn sie vom User aufgerufen werden, in einer anderen App (Internet Explorer) geöffnet werden 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 passiert. In Zeile 3 werden die Infoboxen und die Twitternachrichten herausgefiltert und in den Zeilen 8-13 werden die `<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    }

```

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

4.4. Features

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 Bildwand 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 19). Um eine solche Menüleiste zu erzeugen genügt ein `<div>` Con-

4. Umsetzung der App

tainer 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.

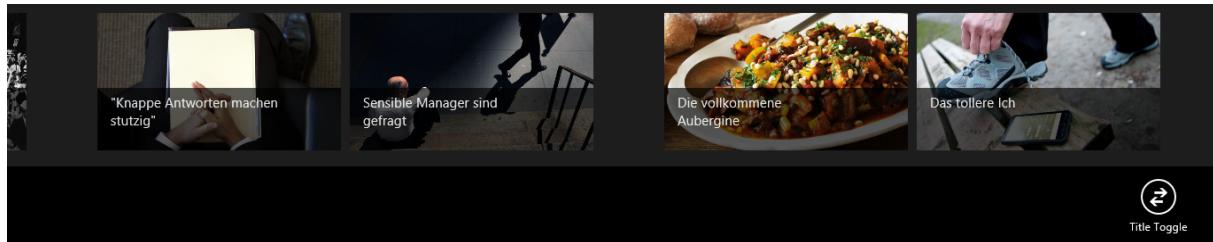


Abbildung 19: Untere Menüleiste mit der Funktion zum Artikeltitel ausschalten.

In der Initialisierungsphase der App wird in der *default.js* Datei der Click-Handler für den Button registriert. D.h. hier wird die Funktion angegeben, welche 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 20 zu sehen.

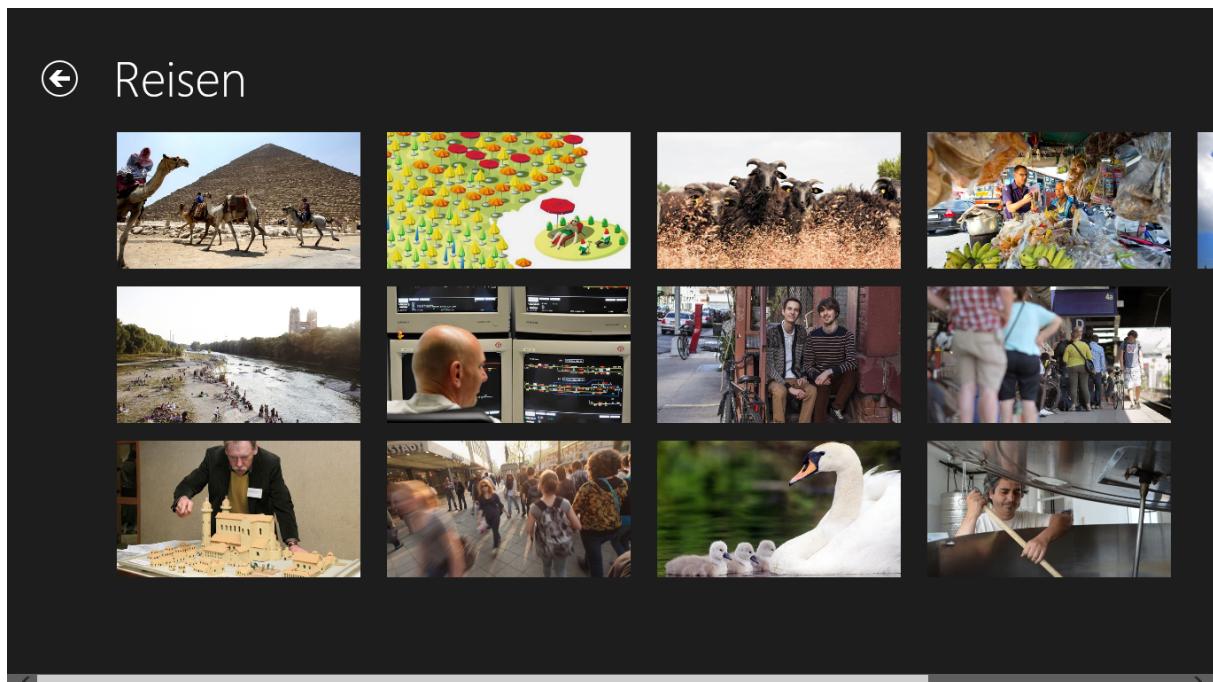


Abbildung 20: Das Reisen Ressort ohne Artikeltitel.

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 die Entwicklungsmöglichkeiten jedoch weiter auszu reizen 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“. Ist die Schriftgröße z.B. auf der größten Stufe eingestellt, wird im Menü, der größer Button nicht mehr dargestellt. Abbildung 21 zeigt die verschiedenen Stati, die das Schriftgrößenmenü haben kann.

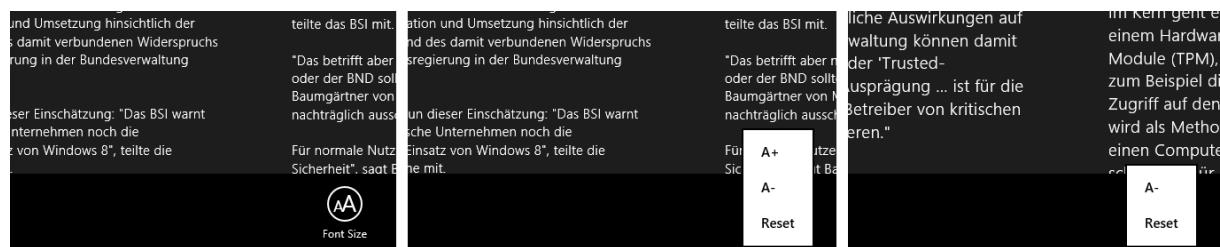


Abbildung 21: links: Schriftgröße Button im unteren Menü | mitte: Das Flyout-Menü mit allen Schaltflächen. | rechts: Das Flyout-Menü beim Aufruf wenn schon die größte Größe eingestellt ist.

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 nur 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 22). 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 Hubansicht „gezoomt“. Der User hätte dann die Ansicht wie sie Abbildung 16 auf Seite 24 zeigt. Das bedeutet es 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.

4. Umsetzung der App

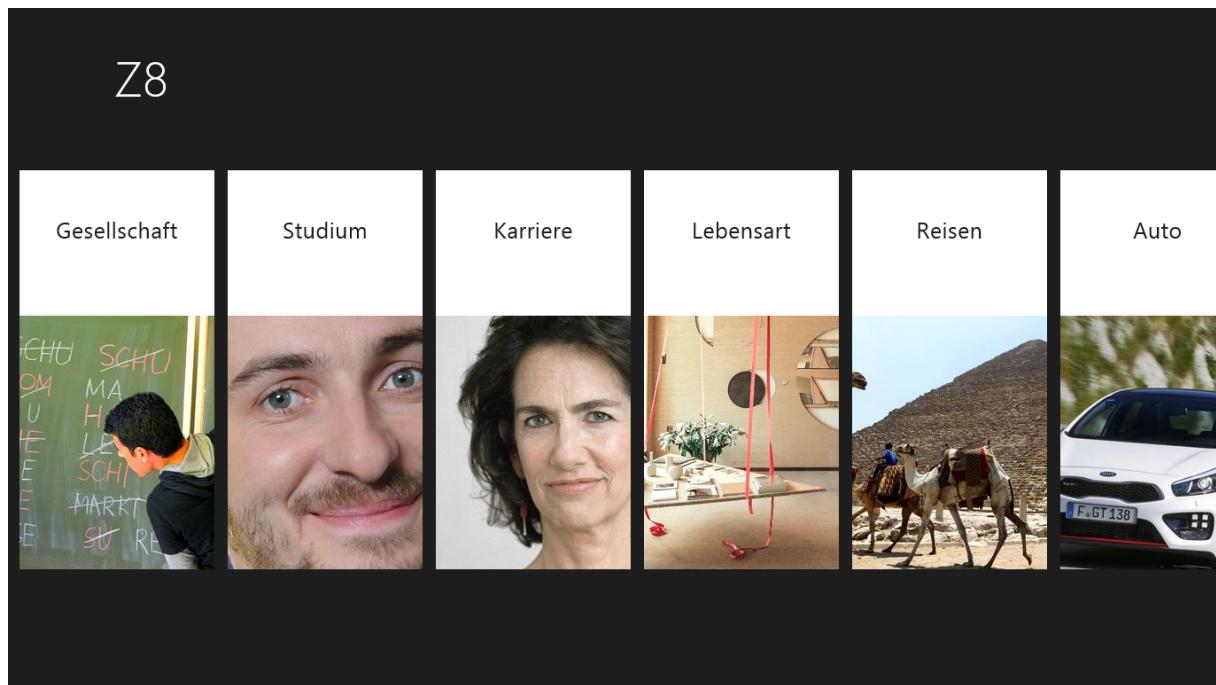


Abbildung 22: Die semantic Zoom Ansicht.

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 ist dieser Zoomstatus initial auf `false`, da der User die Hubansicht als Einstiegs-punkt in die App haben soll (nicht die semantic Zoom Ansicht).

```

1 <div class="semanticZoom" data-win-control="WinJS.UI.SemanticZoom" data-win-
2   -options="{ zoomFactor: 0.5, initiallyZoomedOut: false }" style="height:
3     100%">
4   <div class="groupeditemslist win-selectionstylefilled" aria-label="List
5     of groups" data-win-control="WinJS.UI.ListView" data-win-options="{
6       selectionMode: none}"></div>
7   <div class="groupeditemslistZoomOut groupeditemslist" aria-label="List
8     of groups" data-win-control="WinJS.UI.ListView" data-win-options="{
9       selectionMode: none}"></div>
10 </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 stand, wurde sich zunächst mit der Windows 8 App Entwicklung im Allgemeinen vertraut gemacht, da bis zu diesem Zeitpunkt keinerlei Windows 8 spezifischen Entwicklungskenntnisse vorhanden waren. Es wurden einige Beispiele ausprobiert um ein Gefühl für die Entwicklungsumgebung 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. Stattdessen war am Anfang das nur das Ziel die ZEIT ONLINE Daten in irgendeiner Weise in die App zu bekommen und sie darzustellen. In einer der ersten Versionen der App wurden z.B. nur die Artikeltitel der Homepage angezeigt. Nachdem die Datenanbindung stand, konnte mit dem Entwickeln der richtigen App begonnen werden.

Es wurde zunächst viel recherchiert und in den Microsoft Richtlinien und Styleguides gestöbert um herauszufinden, wie nach Microsoft Sichtweise eine Nachrichten-App auszusehen hat. Dabei war immer wieder die Rede von der Rastervorlage, welche sich für solche Zwecke eignen sollte. 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.

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 angelegt⁸ und die App unter Versionskontrolle gestellt.

Anschließend wurde sich um die Artikelansicht gekümmert. Es wurden die Bildunterschriften und die Copyrights positioniert. Ein weiteres Problem, welches nur bei Entwicklung mit HTML5/JavaScript auftritt war, dass nicht direkt auf die Scrollposition der Hubansicht zugegriffen werden konnte. Deswegen landete der User immer wenn er in die Hubansicht wechselte wieder ganz am Anfang. Da dies nicht zufriedenstellend war, wurde die Funktionalität manuell nachgerüstet, indem der Index

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

4. Umsetzung der App

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 sobald die ListView geladen ist wird die Animation ausgeblendet.

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 ist, konnte einfach der ganze Text, samt Zwischenüberschriften übernommen werden. Das einzige was noch herausgefiltert werden musste sind Infoboxen und Twitternachrichten, die in den Artikel eingebaut sein könnten. 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, ganz abschalten geht nicht. Zuletzt wurden noch Splashscreen und Logos in verschiedenen Größen hinzugefügt.

Es sind hier nicht alle kleinen Tweaks und Fixes an der App beschrieben. Des weiteren sind auch nicht alle aufgetretenen Probleme erwähnt. Es ist lediglich ein grober Überblick über den konkreten Ablauf des Entwicklungsprozesses.

5. Evaluierung

Bei der Evaluierung soll herausgefunden werden, inwieweit die App intuitiv benutzbar ist und ob ein alternativer Ansatz zum Konsumieren von Nachrichten Sinn macht und wie er ggf. funktionieren könnte. Es wird hierzu eine Art der Fokusgruppenevaluierung angewandt. Das bedeutet, es wird am Ende keine belastbaren statistischen Aussagen geben, sondern eher einen Trend zeichnen und einzelne Meinungen der Teilnehmer darstellen.

5.1. Fokusgruppe

Die Bezeichnung „Fokusgruppe“, die hier verwendet wird, steht in diesem Fall für eine kleine Gruppe von Testpersonen, im Gegensatz zu einer vollwertigen statistischen Auswertung. Es findet keine gemeinsame Diskussion statt, stattdessen muss jeder Teilnehmer einige Aufgaben ausführen und ein paar Fragen im Interview beantworten.

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 wurde darauf geachtet, dass sich die Personen möglichst vom Beruf, Geschlecht und Alter her unterscheiden, um eine heterogene Gruppe zusammenzustellen und somit möglicherweise verschiedene Auffassungen und Herangehensweisen zu beobachten.

5.2. Ablauf

Der Ablauf der Evaluierung setzt sich grob aus drei Abschnitten zusammen. Im ersten Teil soll überprüft werden inwieweit die Testpersonen, allein vom Betrachten des Teaserbilds eines Artikels, auf den groben, oder ggf. auch auf den genauen Inhalt des Artikels schließen können. Hierzu werden den Testpersonen zwei Ressorts in der Hubsicht, bei ausgeschalteten Titeln gezeigt. Zum einen, ein Das Politik-Ressort mit vorwiegend tagesaktuellen Nachrichten und zum anderen das Wissen-Ressort, welches auch zeitlose Artikel beinhalten kann. Abbildung 23 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. Au-

5. Evaluierung

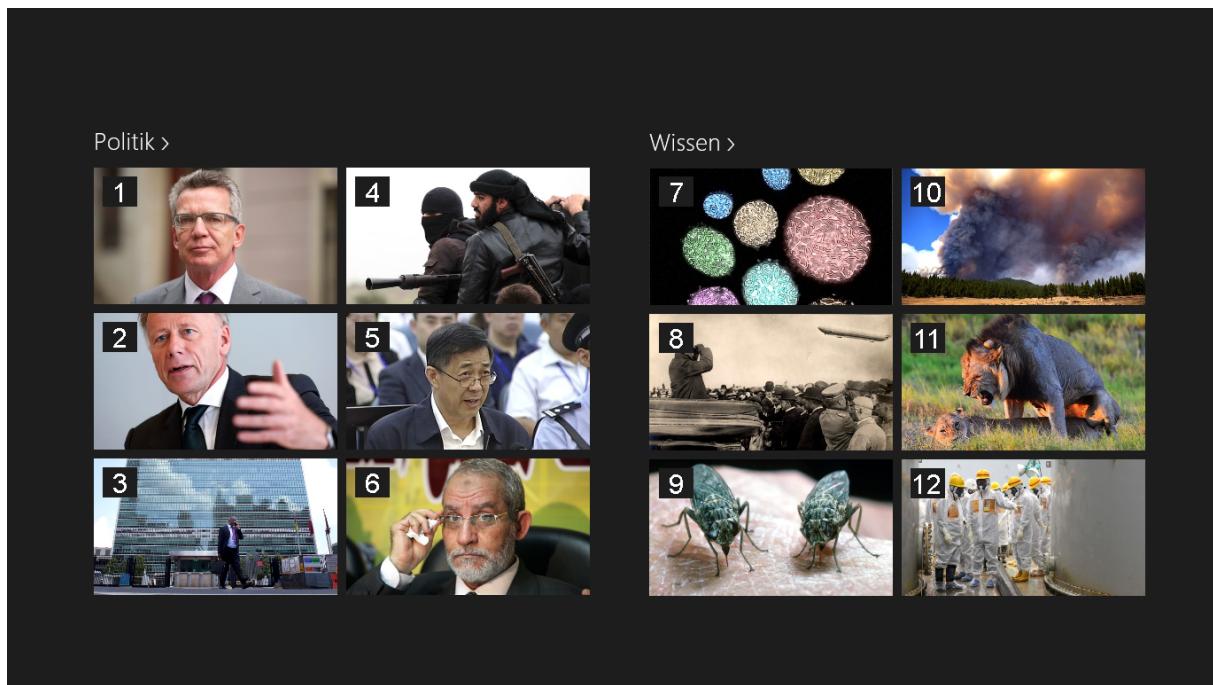


Abbildung 23: Die Testbilder aus dem Politik- und Wissen-Ressort.

gust 2013 auch auf der ZEIT ONLINE Webseite zu finden waren. Die Testpersonen sollen jedes Bild einzeln durchgehen und zunächst eine Einschätzung zum groben Themengebiet geben und anschließend wenn möglich, so genau wie möglich beschreiben, um was 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 groben Fragen aufgelistet, welche im Einzelfall durch Nachfragen leicht abgeändert werden können.

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 vermutlich direkt 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, das 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 Bereiche der App (die 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. Das bedeutet, wenn dieses Navigationselement von einer App benutzt wird, sollte gezielt darauf hingewiesen werden,

5. Evaluierung

da die Möglichkeit eines semantic Zooms, gerade für Windows 8 Einsteiger, oft nicht sofort 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 gewisses Basiswissen, was tagesaktuelle Geschehnisse betrifft, verfügen.

Im ersten Teil der Evaluation, sollten die Testpersonen versuchen herauszufinden, welcher Inhalt sich hinter den Teaserbildern von Abbildung 23 verbergen könnte. Hierzu sollte zunächst ein grobes Themengebiet genannt werden und anschließend so genau wie möglich versucht werden, herauszufinden, 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 78,3% grob, und 35% auch genauer richtig zugeordnet. Das am häufigsten richtig zugeordnete Bild, ist Bild 10. Es wurde von allen Testpersonen grob als auch genau richtig erkannt. Dicht gefolgt Bild 5, welches alle Testpersonen grob, sowie 4 Testpersonen genauer richtig zuordnen konnten. 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.

Die Testpersonen wurden im Interview gefragt, wie sie an diese Aufgabe herangegangen sind. In einem Punkt waren sich alle Testpersonen einig. Wenn man z.B. eine Person erkennt, muss man über andere Nachrichtenkanäle schon vorher etwas über diese Person gehört haben um den Inhalt genauer zuordnen zu können. TP1 meint, bei Bild 1 sei es sehr schwer etwas genaues sagen zu können, da diese Person zu verschiedenen Themen in Frage kommt. 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 versuchen konnten zu raten. 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?“.

5. Evaluierung

Es bleibt noch zu klären ob ein solcher Modus sinnvoll ist und inwieweit der User durch den *nur Bilder* Modus neugierig gemacht wird oder ob er ihn eher in seinem Nachrichtenkonsumfluss stört und hemmt. TP 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 und ist generellt. 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 Entdeckertrieb 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.

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