Hochschule Anhalt

Fachbereich Informatik und Sprachen

Bachelorarbeit

Anpassung der Halle-App nach Microsoft-Designstandards

Zur Erlangung des Grades: Bachelor of Science

Vorgelegt von: Carsten Lutz
Erlenweg 16
06120 Halle (Saale)
carsten.lutz@t-online.de

Studiengang: Angewandte Informatik - Digitale Medien und Spieleentwicklung
Fachsemester: 7
Matrikelnummer: 4058034

Erstgutachter: Prof. Dr. Stefan Schlechtweg-Dorendorf
Zweitgutachter: Prof. Dr. Alexander Carôt

Abgabedatum: 08.01.2018

Inhaltsverzeichnis

1. Einleitung

Mit Windows 10 hat Microsoft die Laufzeitumgebung „Universelle Windows Plattform (UWP)“ eingeführt. Die neuen darauf basierenden Apps von Microsoft bieten viele neue Möglichkeiten für die Entwicklung und Gestaltung.

Die IT-Consult Halle GmbH (ITC) als IT-Dienstleister der Stadt Halle (Saale) entwickelte bereits die Android- und IOS-Version einer mobilen-App, der Halle-App.
Eine Windows-Version der Halle-App in funktionaler und optischer Entsprechung liegt als Prototyp bei ITC vor.

Die Halle-App bietet einen zusätzlichen Online-Auftritt der Stadt Halle neben der Website „www.halle.de“.
Nutzer dieser App sind einerseits Bewohner der Stadt Halle (Saale), die hier die wichtigsten Informationen zum Leben und Wohnen in Halle finden können und anderseits Besucher, die sich zu den wichtigsten Events, Gaststätten und Ausflugszielen informieren können.

Im Verlauf dieser Arbeit erfolgt die Beschreibung des Prozesses, wie der Prototyp der Windows-Version der Halle-App an die Microsoft-Design-Standards angepasst wird. Die Umsetzung der Aufgabe erfolgte in enger Abstimmung und durch Unterstützung der ITC.

Zuerst soll ein theoretischer Überblick in die Thematik der UWP-Apps einführen. Im Besonderen werden die damit verbundenen Geräteklassen und die zugehörige Programmierung erläutert.

Im weiteren Verlauf erfolgt eine Analyse des aktuellen Windows Prototypen der Halle-App. Zusätzlich werden die Konfliktpunkte der aktuellen App mit Bezug zu den Microsoft-Designstandards aufgezeigt.
Im Anschluss wird die Umsetzung der geplanten Änderungen erörtert und die Ergebnisse werden dargestellt.

Am Ende wird ein Resümee gezogen und ein Ausblick auf die Weiterentwicklung der App und der zukünftigen Einsatzmöglichkeiten der UWP-Technologie gegeben.

Diese Arbeit und der angepasste Prototyp bilden die Grundlage für die Weiterentwicklung der App für die ITC.

2. UWP-Apps

Apps der Universellen Windows Plattform (UWP-Apps) sind Anwendungen, die in der Laufzeitumgebung „Universelle Windows Plattform (UWP)“ unter Windows 10 ausgeführt werden können.

2.1. Universelle Windows Plattform

Die UWP wurde mit Windows 10 eingeführt. Sie ist die Weiterentwicklung der Windows-Runtime, welche die Umgebung für Windows Store-Apps unter Windows 8 war.
Unter Windows-Runtime war es erstmals möglich, Apps auf der gleichen Codebasis für unterschiedliche Geräteklassen, wie zum Beispiel Desktop-PC und Mobile-Geräte, zu entwickeln.
In der neuen Universellen Windows Plattform können nun Anwendungen für alle Windows 10-Geräte (PC, Tablet, Smartphone, Xbox, HoloLens, Surface Hub) programmiert werden.
Sie besitzen einen API-Satz, ein App-Paket und ein Store für alle Plattformen [Microsoft 1].
Abbildung 1 zeigt dieses noch einmal zusammengefasst.
Da man auf allen Geräten die gleiche App nutzen kann, ist es die Idee von Microsoft, dass der Benutzer nun wählen kann, mit welchem Windows-Gerät er eine Aufgabe am einfachsten und effizientesten lösen kann.

Abbildung 1 Universelle Windows Plattform

2.1.1. Windows Geräte

In diesem Abschnitt werden die verschiedenen Windows 10 Geräte aufgelistet und erläutert.

Geräteklasse:

Die verschiedenen Geräte von Microsoft werden in Geräteklassen eingeteilt.
Diese sind:

- PCs
- die Mobile-Klasse (Tablets, Smartphones)
- Xbox‘
- Surface Hubs
- HoloLens‘
- Devices oder auch als IoT-Klasse (Internet of Things) bezeichnet

Zur **PC-** (und Laptop-) Klasse gehören viele verschiedene Geräte mit unterschiedlichen Bildschirmen. Zu erwarten ist aber mindestens eine Bildschirmgröße von 13 Zoll [Microsoft 2]. Eingaben erfolgen normalerweise über Maus und Tastatur.

Die **Mobile Klasse** umfasst Smartphones mit einer geringeren Bildschirmgröße von ca. 4 bis 6 Zoll und Tablets mit 7 bis 12 Zoll [Microsoft 2]. Die Eingabe erfolgt meist über Touch. Die meisten Apps werden hauptsächlich auf Mobilgeräten genutzt.

Eine **Xbox** wird an ein TV-Gerät angeschlossen. Diese Bildschirme sind üblicherweise mindestens 24 Zoll groß [Microsoft 2]. Das Besondere ist die Eingabe, die hauptsächlich über einen Controller erfolgt. Hauptsächlich wird die Xbox zum Spielen verwendet, aber auch die anderweitige Nutzung durch andere Apps ist möglich.

Ein **Microsoft Surface Hub** ist ein Großbild-Device für Videokonferenzen und Besprechungen [Wiki 1]. Es gibt die Geräte mit Bildschirmgrößen von 55 Zoll und 84 Zoll [Microsoft 2]. Die Eingabe erfolgt meist über Touch oder Sprache.

Die **HoloLens** ist eine Mixed-Reality Brille von Microsoft. Anders als bei Virtual-Reality-Brillen, die durch 2 Bildschirme eine virtuelle Welt darstellen, schaut der Benutzer durch transparente Bildschirme. Durch projizierte Lichtpunkte werden interaktive 3D-Projektionen in der direkten Umgebung dargestellt. Die HoloLens benötigt dazu keine zusätzlichen Geräte wie PC oder Smartphone und funktioniert autonom. Die Interaktion und Eingabe erfolgt dabei über Gesten, Sprache und Kopfbewegungen [Wiki 2].

Internet der Dinge (englisch: internet of things) bedeutet, dass physische und virtuelle Geräte immer weiter miteinander vernetzt werden. So können IoT-Geräte Sensoren oder Verbindungen mit physischen Objekten besitzen, um Daten zu erfassen und auf mögliche Szenarien reagieren. Zum Beispiel werden solche Geräte in der Medizin verwendet, um Patienten zu überwachen.
Meistens kann das Gerät nur eine Anwendung ausführen, um eine Funktion zu erfüllen, für die es entwickelt wurde. So müssen Anwendungen auch nur die wichtigsten Inhalte darstellen und hauptsächlich funktional sein. Ein Bildschirm oder eine Eingabe von außen sind nicht zwingend erforderlich [Wiki 3].

Gerätefamilien:

Die verschiedenen Geräteklassen werden in Gerätefamilien eingeteilt. Eine Familie wird durch die Systemmerkmale, Verhaltensweisen und Programmierschnittstellen (API), die in einer Geräteklasse erwartet werden können, identifiziert. Eine API ist ein Softwareteil zur Anbindung des Programms an das System auf Quelltext-Ebene. Jede Geräteklasse hat eigene Merkmale, wie zum Beispiel verschiedene Eingabemethoden und spezielle Funktionen, die bestimmte Schnittstellen benötigen [Microsoft 1]. Abbildung 2 gibt eine Übersicht über die Gerätefamilien.

Die Gerätefamilie definiert die Menge von APIs, die für die jeweilige Klasse benötigt werden. Untergeordnete Familien fügen weitere spezifische APIs hinzu, die nur für die jeweiligen Geräte nutzbar sind. Durch diese Schnittstellen können zum Beispiel Apps für die mobile Klasse auf Telefonfunktionen zurückgreifen. Das Konzept der Gerätefamilien spielt hauptsächlich für die Programmierung einer App eine Rolle [Microsoft 1].

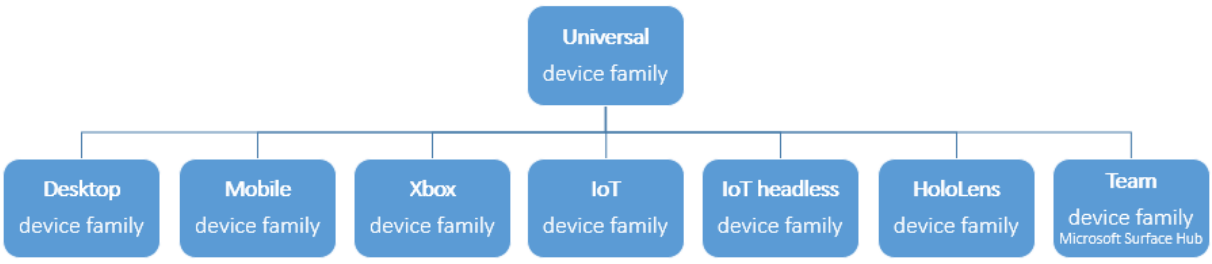

Abbildung 2 Gerätefamilien

Die **universelle Gerätefamilie** nutzt hauptsächlich nur universelle APIs, sodass eine App auf jedem Gerät funktionieren kann. Dennoch können bedingt spezielle APIs für untergeordnete Gerätefamilien genutzt werden, dafür muss der Entwickler diese für eine bestimmte Klasse definieren.
Die Benutzeroberfläche (UI) muss hochgradig adaptiv sein, da sie für alle Bildschirmgrößen angepasst werden muss. Zusätzlich müssen alle Möglichkeiten der Eingabe, wie zum Beispiel Tastatur, Touch, Stift, realisiert werden [Microsoft 1].

Untergeordnete Gerätefamilien benutzen alle universellen APIs und zusätzlich die der familienspezifischen Geräte. Die UI muss nur angemessen adaptiv sein, da die zu erwartenden Bildschirmgrößen der Geräte bekannt sind, für die eine App entwickelt wird. Die Oberfläche muss sich nur an verschiedene Größen innerhalb einer Klasse anpassen können. Somit sind auch nur die jeweiligen erforderlichen Eingabemöglichkeiten umzusetzen. [Microsoft 1].

Die **IoT-Gerätefamilie** spielt eine besondere Rolle. Die UI kann sehr speziell sein, da eine App meistens nur für ein bestimmtes Gerät entwickelt wird, welches eine bestimmte Funktion erfüllen muss. Bildschirme oder Eingabemöglichkeiten sind nicht zwingend erforderlich [Microsoft 1].

Für den Entwickler ergeben sich zwei Möglichkeiten,
Zum einen könnte er die maximale Verbreitung in Bezug auf die verschiedenen Geräteklassen mit seiner App erreichen und diese für die universelle Gerätefamilie ausrichten.
Damit wäre er aber bezüglich der APIs eingeschränkt und hat einen hohen Aufwand bei der Entwicklung.
Die Alternative besteht darin, dass er seine Auswahl auf eine oder mehrere Gerätefamilien begrenzt, wodurch die Funktionalitäten der Geräte durch speziellere APIs besser genutzt werden können und der Aufwand der Anpassung geringer ist [Microsoft 1].

2.1.2. Programmierung

Nach der Entscheidung über die Gerätefamilie muss zusätzlich die Wahl für die passende Entwicklungsumgebung und Programmiersprache vom Entwickler getroffen werden.

UWP-Apps werden standardmäßig in Visual Studio, von der Firma Microsoft, programmiert. Die erste Version in der dies möglich war ist Visual Studio 2015.

Für die Entwicklung werden von Visual Studio mehrere Programmiersprachen unterstützt. Zur Umsetzung der Microsoft Adaption der Halle-App wurde zum Beispiel C# für die Funktionen und XAML für die Oberfläche genutzt. XAML (Extensible Application Markup Language) ist eine sogenannte Beschreibungssprache, die zur Gestaltung von grafischen Benutzeroberflächen verwendet wird.

Unabhängig von der Halle-APP werden hauptsächlich 3 verschiedene Sprachen für die UI- und Oberflächengestaltung genutzt. Diese sind XAML, DirectX und JavaScript. Darauf aufbauend werden in den CodeBehind-Dateien Funktionen und Ereignisse (z.B. Buttonklicks) der App umgesetzt. Für eine XAML gestaltete Oberfläche können C#, Visual Basic und C++ zur Programmierung der dahinterstehenden Funktionen verwendet werden. DirectX wird hauptsächlich für Multimedia Anwendungen, insbesondere für Spiele, verwendet und mit C++-Code ergänzt. Mit Hilfe von HTML können JavaScript basierende Apps ihre Funktionen nutzen [Microsoft 3].

2.2. Besonderheiten gegenüber Android/ iOS

Der größte Unterschied von UWP-Apps zu Android- und IOS-Apps ist, dass die UWP-Apps nicht ausschließlich für mobile Geräte entwickelt werden. Durch universelle APIs ist es möglich, dass jedes Windows 10-Gerät dieselbe App nutzen kann. Darauf aufbauend gibt es mehrere Unterschiede in der Programmierung und Gestaltung.
Durch die Nutzung von effektiven Pixeln, können die Elemente der Benutzeroberfläche (UI) an die Geräte und auf die verfügbaren Pixel der zugehörigen Bildschirme angepasst werden. Effektive Pixel basieren auf einem speziellen Skalierungsalgorithmus, der den Abstand zum Bildschirm und Bildschirmdichte berücksichtigt, um die wahrgenommene Größe zu optimieren So ist ein reaktionsfähiges Design möglich, sodass UI-Elemente komplett verändert werden können, je nach Gerät auf dem die App verwendet werden soll [Microsoft 4].

Der Microsoft-Store ist einheitlich für alle Geräteklassen, also identisch auf dem PC oder dem Smartphone. Alle Apps können hier zentral verwaltet werden. Zusätzlich stellt der Store genau die richtige Version einer App für das benutzte Gerät bereit [Microsoft 1].

UWP-Apps werden zusätzlich im AppX Paketformat verpackt. Dieses stellt zusätzlich einen vertrauenswürdigen Installationsmechanismus bereit. So können Apps problemlos über den Store verteilt werden [Microsoft 1].

2.3. UWP Designstandards

Im folgenden Abschnitt werden die Designstandards und -richtlinien, die Microsoft in ihrer Dokumentation über UWP-Apps erwähnt, erläutert.

2.3.1. Responsive Design

„The control which designers know in the print medium, and often desire in the web medium, is simply a function of the limitation of the printed page. We should embrace the fact that the web doesn't have the same constraints, and design for this flexibility. "[Allsopp]

„Im Web- oder App-Design ist der Trend: eine Version für alle [Geräte], aber dabei flexible Anpassungen an die spezifischen Gegebenheiten. Responsive Design bezeichnet die Fähigkeit einer Website [oder App] automatisch auf die jeweiligen Herausforderungen reagieren zu können" [Rohles].

Im klassischen Printdesign hat das Ausgabemedium, zum Beispiel eine Tageszeitung, eine feste Größe. So können Designer diese starr und unveränderlich gestalten. Dies funktioniert im Web-, bzw. App-Design nicht mehr. Besonders bei UWP-Apps muss die Oberfläche auf verschiedene Gegebenheiten reagieren können und je nach Gerät eventuell das Design ändern.
Jeder Gerätetyp weist verschiedene Arten von Eingaben, Auflösungen, DPI-Dichte und andere Merkmale auf. So muss eine App an die diversen Merkmale für jede Bildschirmgröße angepasst werden. Es gibt mehrere neue Mittel und Wege das umzusetzen. So bietet die Universelle Windows Plattform Steuerelemente, Layout-Bereiche und Tools, mit denen die Benutzeroberfläche angepasst werden kann.

Aufgrund der speziellen UI können Steuerelemente wie Schaltflächen und Schieberegler automatisch angepasst werden. Trotzdem muss eventuell das Design je nach App angepasst werden.
Im Zusammenhang damit stehen die reaktionsfähigen Designtechniken, mit denen die App an bestimmte Bildschirmbreiten optimiert wird. Ein Beispiel ist, dass der kleine meist im Hochformat genutzte Smartphone-Bildschirm dieselbe App darstellen soll, wie der große, querformatige PC-Bildschirm. Jede Designänderung muss vom Entwickler selbst angepasst werden. Vor allem Apps für die universelle Gerätefamilie benötigen viele verschiedene Designs, die an den Bildschirm angepasst werden müssen.

Reaktionsfähige Designtechniken sind das Ändern der Position, Ändern der Größe, Neuanordnen, Einblenden, Ersetzen oder das Ändern der Architektur [Microsoft 5].

- **Ändern der Position:**
  Beim Ändern der Position werden UI-Elemente je nach Bildschirmgröße verschoben, z.B. können im Querformat UI-Elemente nebeneinander angezeigt werden, während diese im Hochformat untereinander mit einem Bildlauf dargestellt werden.

Abbildung 3 Reaktive Designtechniken: Ändern der Position

- **Ändern der Größe:**

  Das Ändern der Größe kann verwendet werden, um Inhalt wie Text besser darzustellen. Dieser wird an das Format angepasst.

Abbildung 4 Reaktive Designtechniken: Ändern der Größe

- **Neuanordnen:**

  Beim Wechsel des Bildformates ist es möglicherweise sinnvoll, größere Inhaltselemente zu verwenden oder neue Spalten hinzuzufügen. Dadurch werden Inhalte neu angeordnet.

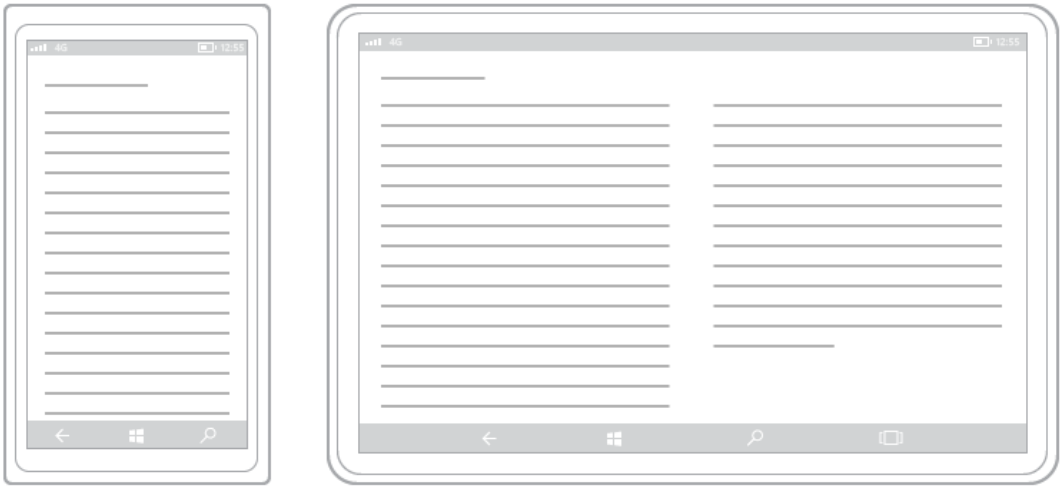

Abbildung 5 Reaktive Designtechniken: Neuanordnen

- **Einblenden:**

  Wenn ein Gerät mögliche Funktionen nicht besitzt, so müssen auch die Benutzeroberflächenelemente dafür nicht angezeigt werden. Es können aber auch zusätzliche Steuerelemente bei größeren Bildschirmen angezeigt werden, die beim kleinen Bildschirm über ein zusätzliches Menü geöffnet werden müssen.

Abbildung 6 Reaktive Designtechniken: Einblenden

- **Ersetzen:**

  Unter Umständen sind manche Navigationsmenüs nicht für alle Bildschirmgrößen geeignet, deshalb kann es nötig sein, diese durch andere zu ersetzen.

Abbildung 7 Reaktive Designtechniken: Ersetzen

- **Ändern der Architektur:**

  Manche Funktionen können möglicherweise nicht richtig übertragen werden, sodass die Architektur der App angepasst werden muss. So sollte das gesamte Design der App überarbeitet und neugestaltet werden, damit diese für bestimmte Geräte optimal angezeigt werden.

Abbildung 8 Reaktive Designtechniken: Ändern der Architektur

Universelle Eingabe

UWP besitzt universelle Steuerelemente die auf jedem Windows-Gerät funktionieren, diese können Eingaben über die Maus, Tastatur, Fingereingaben, über einen Stift oder einem Controller verarbeiten. Diese Steuerelemente basieren auf der allgemeinen Eingabebehandlung, die je nach Art des Gerätes angepasst werden.

Tools

Microsoft stellt neue Tools und Funktionen unter Windows 10 zur Verfügung, die direkt das Responsive Design unterstützen.

Für den Bereich Adaptive UI wurde das neue Relative Panel implementiert. In diesem Panel werden nun nicht mehr alle Positionen der untergeordneten Elemente definiert, sondern nur noch die Beziehungen untereinander. So kann die UI besser an alle Bildschirmgrößen und Auflösungen angepasst werden, da es durch seine Eigenschaften den Prozess der Neuanordnung von Elementen vereinfacht [Microsoft 1].

Eine wichtige Rolle spielen hier die so genannten Zustandsauslöser. Diese verändern ab einem bestimmten Schwellenwert der Bildschirmgröße das Design der Oberfläche. Es können zum Beispiel neue Inhalte im selben Fenster angezeigt werden, wenn der festgelegte Wert überschritten wird. Die Zustandsauslöser können auch im direkten Zusammenhang mit den reaktiven Designtechniken verwendet werden. Im Unterschied dazu werden hier aber nur die Oberflächen innerhalb einer Geräteklasse an verschieden große Bildschirme angepasst [Microsoft 1].

2.3.2. Strukturierung der Anwendung

Die drei wichtigen Faktoren für die Gestaltung der Struktur sind die Navigations-, Befehls- und Inhaltsemente.

Navigation:

In UWP-Apps basiert die Navigation auf einem flexiblen Modell aus Navigationsstrukturen, Navigationselementen und Funktionen auf Systemebene. Zusammen soll eine intuitive Benutzererfahrung beim Navigieren zwischen Seiten und Inhalten gewährleistet werden. Um dies zu erreichen, müssen die drei Grundprinzipien Einheitlichkeit, Einfachheit und Konsistenz erfüllt werden [Microsoft 6].

Der Benutzer erwartet bestimmte Elemente immer an der gleichen Position. Zum Beispiel befindet sich die Navigationsleiste in der Regel immer oben bei einer App für Mobilgeräte. Dies sollte einheitlich in jeder App und in jeder Ansicht sein.
Die Navigation sollte so einfach wie möglich gehalten werden. In den meisten Fällen sind zwei Navigationsebenen ausreichend. So wird normalerweise eine Navigation auf oberster Ebene und eine Hierarchieebene verwendet. Durch zu viele Ebenen wird die Navigation kompliziert und für den Benutzer ist eine tiefe Hierarchie unübersichtlich [Microsoft 6].

Zusätzlich sollten in jeder Ebene ungefähr drei bis maximal 6 Navigationselemente verwendet werden.
Es ist wichtig, dass die Navigation im Kontext zum Inhalt steht und (Navigations-) Ziele schnell erreicht werden, um eine konsistente und saubere Interaktion zu erreichen. Zum Beispiel könnte eine Koch-App einen schnellen Zugriff auf die Einkaufsliste bereitstellen. Dabei sollte das sogenannte „Pogo Sticking“ vermieden werden, In dem der Benutzer immer wieder erneut eine Ebene nach oben navigieren müsste, um zusammengehörige Inhalte erfassen zu können [Microsoft 6].

<u>Die richtige Navigationsstruktur:</u>
Eine für den Benutzer sinnvolle Navigationsstruktur ist für eine intuitive Funktionalität der Navigation wichtig. Zu beachten ist, dass die meisten Apps oft aus mehreren Seiten von Inhalten und Funktionen bestehen sollen. So muss die geeignete Struktur gewählt werden. Es werden oftmals Navigationsstrukturen in einer Hierarchie oder als Peers organisiert [Microsoft 6].

- **Hierarchie:**

  Die Seiten sind typischerweise in einer Baumstruktur organisiert. So hat jedes untergeordnete Element ein übergeordnetes Element. Übergeordnete Elemente können mehrere Untergeordnete besitzen. Typischerweise wird diese Struktur genutzt, wenn die Seiten in einer bestimmten Reihenfolge durchlaufen werden sollen und eine klare Beziehung zwischen übergeordneten und untergeordneten Elementen herrscht.

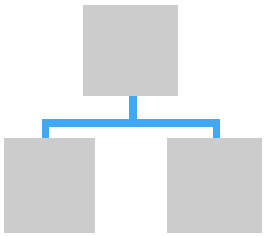

Abbildung 9 Hierarchischen Navigation

- **Als Peers:**

  Die Seiten sollen nebeneinander existieren und können in beliebiger Reihenfolge angezeigt werden. Wenn keine klaren Beziehungen zwischen über- und untergeordneten Elementen besteht sollte diese Struktur verwendet werden.

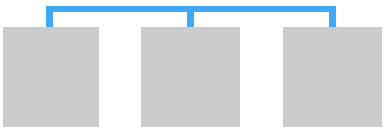

Abbildung 10 Navigation als Peers

<u>Geeignete Navigationselemente:</u>

Mithilfe von Navigationselementen können vom Benutzer Inhalte ausgewählt werden, die in der App angezeigt werden sollen. Für die Struktur ist zu beachten, dass die Elemente ebenfalls Platz auf der Oberfläche benötigen.

- **Master/Detail (Hierarchisch):**

  Es wird eine Liste (Masteransicht) der Elementübersichten angezeigt. Wenn ein Element ausgewählt wird, wird der entsprechende Detailbereich angezeigt. Master/Detail wird häufig verwendet, wenn der Benutzer häufig zwischen untergeordneten Elementen wechseln wird und hierarchische Beziehungen bestehen.

- **Navigationsbereich (Peer-to-Peer):**
  Es wird eine Liste mit übergeordneten Seiten angezeigt. Häufig wird es verwendet, wenn der Benutzer nicht so häufig zwischen den Seiten wechselt wird und sich alle Seiten auf einer Ebene befinden sollen.
- **Registerkarten und Pivots (Peer-to-Peer):**
  Es wird eine dauerhafte Liste (Registerkarten) mit Links zu Seiten derselben Ebene angezeigt. Dieses Element empfiehlt sich, wenn der Nutzer häufig zwischen den Seiten wechseln muss und die Navigationsebene nicht mehr als fünf Seiten besitzt.

Befehlselemente:

Befehlselemente dienen der Interaktion mit der Benutzeroberfläche, mit denen der Benutzer Aktionen durchführen kann, wie z.B. E-Mails senden oder ein Formular übermitteln. So müssen die entsprechenden Elemente für die Benutzerinteraktionen ausgewählt werden. Diese sind wichtig für eine intuitive Bedienung der App. So kann der Nutzer die Anwendung ohne größere Probleme und Verwirrungen verwenden. Werden mehrere Befehlselemente benötigt, können diese auf der passenden Oberfläche platziert werden [Microsoft 7].

- **Inhaltsbereich:**
  Werden Befehlselemente im Inhaltsbereich benötigt, so kann man diese direkt neben oder auf Objekten platzieren.
- **Befehlsleiste:**
  Eine Befehlsleiste kann am oberen oder am unteren Bildschirmrand platziert werden. Sie enthält eine Sammlung von mehreren Befehlselementen.
- **Menüs und Kontextmenüs:**
  Befehlselemente können auch in Menüs gruppiert werden, um platzsparend mehrere Optionen unterzubringen.
  Kontextmenüs werden oft an häufig verwendete Aktionen geknüpft, zum Beispiel Anzeige von Befehlen beim Klick auf ein Inhaltsobjekt oder die Möglichkeit zum Kopieren von Text, wenn dieser ausgewählt wird.

Inhalt:

Das Bereitstellen von Zugriff auf Inhalte ist die Hauptaufgabe jeder App. Es wird ein Überblick über die drei wichtigsten Content-Szenarien gegeben [Microsoft 8].

- **Konsum:**
  Konsumorientierte Apps konzentrieren sich hauptsächlich auf die einseitige Erfahrung, bei der der Content nur konsumiert wird. Eine intuitive Navigation durch die Inhaltsseiten muss gewährleistet sein, zum Beispiel Lese-Apps, Musik-Apps, Foto-Viewer. Bei der Erstellung ist zu beachten, dass Navigationsseiten von Content Seiten getrennt werden, damit der Nutzer den Inhalt ohne Ablenkung ansehen kann.
- **Erstellen:**
  Erstellungsorientierte Apps konzentrieren sich hauptsächlich auf die Erstellung oder Bearbeitung von Content. Dafür werden besonders die passenden Befehlselemente benötigt, um eine problemlose Arbeit zu ermöglichen, zum Beispiel Zeichen-Apps, Textbearbeitungs-Apps. Die Verwendung von Navigationselementen sollte minimal sein.
- **Interaktiv:**
  Interaktive Apps stellen eine Mischform aus konsumorientierten und erstellungsorientierten Apps dar. Hauptaufgabe ist es Inhalt wiederzugeben und diesen bearbeiten zu können. Dabei muss ein Gleichgewicht von Navigationselementen, Befehlselementen und Inhaltelementen geschaffen werden, ohne dass dabei die App zu überladen wirkt. So können mehrere separate Seiten für die Funktionen eine Lösung darstellen, zum Beispiel Branchen-Apps, Koch-Apps.

2.3.3. Design der Anwendung

Der zweite wichtige Teil neben der Strukturierung ist die grafische Gestaltung einer UWP-App. Dafür müssen einige Richtlinien eingehalten werden.

Farbe:

Farben unterstützen den Benutzer, sich intuitiv auf den verschiedenen Informationsebenen zurecht zu finden. Bei der Farbwahl ist zu beachten, dass Microsoft sogenannte Akzentfarben für das allgemeine Design eines Geräts bereitstellt. Dies sind 48 speziell zusammengestellte Farbmuster, die alle Oberflächen beeinflussen, zum Beispiel den Hintergrund in Microsofts Kachelmenü. Diese sind in Abbildung 11 dargestellt. Für die Gestaltung des Inhalts einer App, besonders bei Texten oder Symbolen, sollten keine Akzentfarben verwendet werden. Es kann dazu führen, dass die Schrift schwer erkennbar wird, wenn der Nutzer zufällig die gleiche Akzentfarbe für seinen Hintergrund auf seinem Gerät ausgewählt hat [Microsoft 9].

Grundsätzlich wird bei Apps in helles und dunkles Design unterschieden. Der Entwickler hat dabei die Wahl, ob er es selbst festlegt oder das Design sich automatisch an das Gerät des Benutzers anpasst.
Apps mit hellem Design sind hauptsächlich für produktives Arbeiten geeignet. Dies umfasst das Lesen umfangreicher Texte und längeres konzentriertes Arbeiten.
Apps mit dunklem Design sind besonders für das wiedergeben von Bildern und Videos geeignet. Dadurch wird der Kontrast vor allem bei schwacher Umgebungsbeleuchtung verbessert. Das Lesen von Texten steht hier im Hintergrund [Microsoft 9].

Abbildung 11 Windows Akzentfarben

Typografie:

Eine wichtige Funktion von Apps ist das Informieren des Nutzers. Dazu wird die Schrift als visuelle Darstellung von Sprache genutzt. Dazu stellt Microsoft Vorschläge und Richtlinien bereit, um eine übersichtliche Typografie zu gewährleisten.

Die Standardschriftart ist „SegoeUI“, da sie eine breite Palette von Zeichen besitzt und gut lesbar auf unterschiedlichen Displaygrößen ist. Es ist eine humanistische Schriftart mit weicher und ansprechender Optik und organischen offenen Formen. Es wird empfohlen, eine geringe Anzahl von Schriftbreiten und –graden sowie eine klare Hierarchie zu verwenden. Diese basieren auf dem Standardstil für jede Sprache. Der Zeilenabstand soll 125% des Schriftgrades entsprechen. Bei SegoeUI mit 15px wäre dies 18,75PX. Befindet sich eine größere Schrift über einer kleineren, wird der Zeilenabstand mit der größeren berechnet. Es wird empfohlen eine linksbündige Ausrichtung für visuelle Elemente und Spalten mit Schrift zu verwenden. Dadurch werden eine konsistentere Verankerung des Inhalts und ein einheitliches Layout gewährleistet. Kurze Zeilen mit einer geringen Anzahl an Zeichen stören den Lesefluss, da das Auge sich ständig hin und her bewegen muss. Aber zu lange Zeilen können unübersichtlich und verwirrend sein, deshalb liegt die optimale Zeichenanzahl bei 50-60 Zeichen pro Zeile [Microsoft 10].

Symbole:

Es sollen ausdrucksstarke, eindeutige Symbole und keine Metaphern verwendet werden, damit der Nutzer diese schnell und einfach verstehen und nutzen kann.

Bewegungen:

Das Ziel von Bewegungen in einer App ist es, diese lebhafter und realistischer erscheinen zu lassen. Sie sollen den Benutzer dabei unterstützen, sich zu orientieren und Kontextänderungen zu verstehen. Dies wird erreicht, indem Interaktionen mit visuellen Übergängen verbunden werden. Die App erhält mehr Geschwindigkeit und Mehrdimensionalität [Microsoft 11].

Hauptsächlich sollen Bewegungen dem Zweck der App dienen und die Benutzeroberfläche zum Leben erwecken. Dabei ist wichtig, dass der Benutzer durch die Bewegungen visuelles Feedback seiner Interaktionen bekommt. Dadurch wird verständlich aufgezeigt, wie die Ansichten miteinander verbunden sind und der Nutzer durch die App navigieren kann.

Besonders bei größeren Apps mit vielen Ansichten und Oberflächenelementen oder wenn der Benutzer komplexere Aufgaben lösen muss, sind qualitativ hochwertige Bewegungen sehr wichtig. Denn dadurch kann die kognitive Belastung verringert werden, da die Zusammenhänge übersichtlicher werden. Damit wird auch die Wahrnehmung der Benutzerfreundlichkeit gesteigert [Microsoft 11].

Arten von Bewegungen:

- **Verbundene Animationen:**
  Bei verbundenen Animationen wird der Übergang eines Elementes zwischen zwei Ansichten animiert, dadurch wird eine dynamische und ansprechende Navigationsfunktionalität erzeugt. Das Element scheint zwischen den beiden Ansichten weiter zu bestehen, während sich der Inhalt der Oberfläche ändert. Zum Beispiel kann ein Link zu einer anderen Seite in dieser als Überschrift verwendet werden. Das Element fliegt dabei von seinem Standort in der Quellansicht zu dem Ziel in der neuen Ansicht. Durch diesen Effekt wird die Beziehung zwischen den zwei Ansichten hervorgehoben, der für einen dynamischen Übergang sorgt. Der Benutzer kann dadurch den Zusammenhang zwischen den beiden Seiten besser nachvollziehen.
- **Inhaltsübergang:**
  Durch Inhaltsübergangsanimationen können Inhalte eines Bildschirmbereiches verändert werden ohne den Hintergrund zu beeinflussen. Dies wird zum Beispiel angewendet, wenn ein vorhandener Text in einem Container durch einen neuen ersetzt wird, aber die Ansicht die gleiche bleiben soll. Der neue Inhalt wird eingeblendet, während der andere ausgeblendet wird.
- **Hinzufügen und Löschen:**
  Diese Animationen werden zum Beispiel bei Listen verwendet, um Elemente hinzu zu fügen oder zu entfernen. Diese sind aber nur geeignet, Elemente in vorhandenen Containern zu verändern und nicht um neue Listen zu erzeugen.
- **Feedback durch Drücken:**
  Bei „Feedback durch Drücken“ wird der Zeiger animiert, wenn auf ein Element getippt wird, um visuelles Feedback zu liefern. Diese werden hauptsächlich bei Desktopanwendungen genutzt.

Extras:

Folgende Designelemente wurden noch nicht für die Gestaltung von UWP-Apps veröffentlicht, werden aber zukünftig eine wichtige Rolle spielen [Microsoft 12]:

- **Acryl**:
  Mit Hilfe des Acryleffekts können teilweise transparente Texturen erzeugt werden. Es wird eine physische Struktur und Tiefe hinzugefügt, um eine visuelle Hierarchie zu schaffen. Beim sogenannten „In-App-Acryl“ wird der Effekt hauptsächlich auf unterstützende UI-Komponenten wie Navigation oder Steuerelemente innerhalb der App angewendet. So wird eine Tiefenwirkung und Hierarchie innerhalb einer Ansicht geschaffen. Acryl sollte nur in optisch unauffälligen Bereichen wie zum Beispiel bei der Navigation und nicht als durchsichtiger Hintergrund für den Inhaltsbereich genutzt werden, da sonst visuelle Störungen in der Benutzerfreundlichkeit erzeugt werden können.
  Das „Hintergrund-Acryl“ wird vor allem bei Desktop Apps genutzt, da man durch die transparente Schicht Desktophintergrund und andere Fenster noch erkennen kann. So wird eine Tiefe zwischen den einzelnen Ebenen geschaffen. Trotzdem sollten für den Inhaltsbereich einheitliche, undurchsichtige Hintergründe genutzt werden [Microsoft 13].

- **Parallax**:
  Bei Parallax handelt es sich um einen visuellen Effekt, bei dem Vordergrundelemente sich schneller bewegen als solche im Hintergrund. Dadurch wird ein Gefühl der Tiefe, Perspektive und Bewegung geschaffen. Wenn im Vordergrund ein Bildlauf bei einem Text verwendet wird, so wird dieser schneller bewegt als das Hintergrundbild. Dieser Effekt wird mit einem neuen Steuerelement dem Parallax View erzeugt werden können [Microsoft 14].

3. Analyse der Halle-App

In diesem Abschnitt wird die Halle-App des vorhandenen Prototyps hinsichtlich Aufbau und Windows-Designstandards analysiert.

3.1. Menüstruktur

Homemenü:

Abbildung 12 Homemenü Android App

Das Homemenü beinhaltet die Navigation zu den fünf Untermenüs: Rathaus, Stadtinfos, Veranstaltungen, Nachrichten und Baustellenkalender.

Rathaus (Untermenü):

Abbildung 13 Rathausmenü Android App

Das Rathausmenü soll Bürger der Stadt Halle über Dienstleistungen informieren. In einer großen Datenbank kann man über vorsortierte Kategorien bestimmte Dienstleistungen finden und dadurch zuständige Stellen, erforderliche Dokumente, etc. in Erfahrung bringen. Zusätzlich werden hier noch Möglichkeiten zur besseren Kommunikation zwischen Bürgern und der Stadtverwaltung angeboten. Die jeweiligen Buttons führen zu externen Seiten außerhalb der Halle-App.

Stadtinfos (Untermenü):

Abbildung 14 Stadtinfos Menü Android App

Das Stadtinfos-Menü beinhaltet Information, die für einen Besuch von der Stadt Halle wichtig sind. Einerseits wird hier in den oberen drei Menüpunkten die Stadt Halle vorgestellt, andererseits informieren die anderen zum Aufenthalt und ausgewählten Ausflugszielen.

Veranstaltungen (Untermenü):

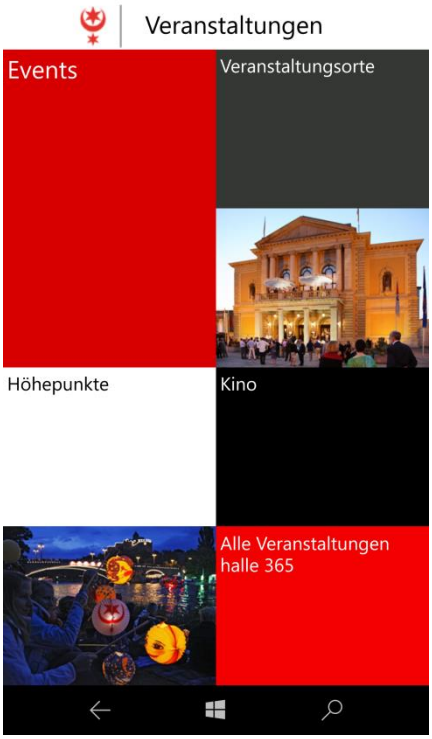

Abbildung 15 Veranstaltungen Menü Android App

Wie der Titel bereits erwähnt, findet man hier Informationen zu Veranstaltungen in und rund um Halle. Unter Events erfährt man über diese in einem nach Kategorien und Datum sortierten Untermenü. Wenn man mehr Informationen zu den einzelnen Veranstaltungsorten haben möchte, navigiert man durch dieses Menü. Ähnlich wie das Dienstleistungsmenü besitzen diese beiden eine große Datenbank im Hintergrund. Die anderen Punkte führen zu verschiedenen Seiten von „halle365“, die alle Informationen für das ganze Menü Veranstaltungen bereitstellen.

Nachrichten (Untermenü):

Im Bereich Nachrichten werden alle Neuigkeiten in bzw. um Halle veröffentlicht.

Baustellenkalender (Untermenü):

Der Baustellenkalender ist eine von den Stadtwerken Halle entwickelte Anwendung, die tagesaktuell alle Baustellen in der Stadt Halle anzeigt. Dazu wird eine Karte von OpenStreetMap verwendet.

3.2. App Struktur

Navigation:

Einheitlich (Navigation an erwarteter Stelle):

Es gibt keine explizite Navigationsleiste in der App. Die Navigation erfolgt über die Kacheln auf dem gesamten Bildschirm. Da die grundlegende Navigation auf zwei Ebenen verteilt ist, ist diese sehr unübersichtlich und verwirrend, dazu kommt noch die Navigation durch die unteren Ebenen.

Einfach (3-6 Navigationselemente, 2 Ebenen):

Wenn man zum Beispiel nach einer bestimmten Dienstleistung sucht, muss man durch vier Ebenen navigieren, um den Inhalt angezeigt zu bekommen. Dies sind zu viele.
Mit Ausnahme des Rathausmenüs haben alle grundlegenden Navigationsseiten (Ebene 1 und 2) maximal 6 Elemente auf einem Bildschirm. Die unteren Navigationsebenen enthalten große Listen mit vielen alphabetisch sortierten Elementen.

Konsistenz (Pogo Sticking vermeiden):

Vor allem wenn man ständig durch die obersten beiden Ebenen der App navigieren möchte, muss man oft die Seiten wechseln. Dies sollte optimiert werden. Die Navigation durch die unteren Ebenen ist sortiert und notwendig, um genau Informationen zum gewünschten Thema zu erhalten.

Befehlsleiste:

Die Halle-App besitzt keine Befehlsleiste.

Inhalt:

Die Halle-App ist konsumorientiert, das heißt, dass die Informationsvermittlung und der Inhalt im Mittelpunkt stehen. Die Navigation zu den Inhalten von „Dienstleistungen“, „Veranstaltungsorten“ und „Events“ erfolgt über mehrere lange Listen von Kategorien, diese sind sehr umfassend aber notwendig, um den möglichen Content darzustellen. Dadurch wird auch erreicht, dass die speziellen Inhaltsseiten ohne Ablenkung von Navigation und in voller Bildschirmgröße gelesen werden können.

3.3. Design

Farbe:

Es ist zu prüfen ob die Farbwahl den Richtlinien von Microsoft entsprechen.

Für die Halle-App wurde ein helles Design verwendet. Dies ist besonders für das Lesen vieler Texte geeignet, so ist es leichter, die Inhalte zu erfassen. Hauptaufgabe der App ist die Informationsvermittlung, so ist es wichtig, dass die Inhalte gut lesbar sind.

Typografie:

Der Prototyp der Halle-App für Microsoft wurde bereits im Visual Studio erstellt, deshalb sind die meisten formalen Punkte bereits voreingestellt. Trotzdem wurde im bereits bestehendem Programm die Schriftart Source Sans Pro verändert, welche ebenfalls für die anderen Versionen für IOs und Android genutzt wurde.

Bewegungen:

Bisher enthält die App keinerlei Bewegungen und Animationen um visuelle Verbindungen, um ansprechende Übergänge zu schaffen.

3.4. Responsive Design

Durch automatische Größenanpassungen kann sich die App an verschiedene Bildschirmgrößen der mobilen Geräteklasse anpassen. Die verschiedenen Elemente sind aber nicht für eine adaptive UI angepasst. Die Größe der Elemente kann zwar automatisch angepasst werden. aber die Positionen sind fest definiert. Besonders bei größeren Bildschirmen würde das dazu führen, dass die Buttons riesig werden und große leere Bereiche entstehen würden. Möglicher Platz würde ungenutzt bleiben. Um das zu verhindern, müsste die Oberfläche der App angepasst werden. Mit Hilfe von reaktionsfähigen Designtechniken kann dies umgesetzt werden. Dazu könnte man entweder das aktuelle Design anpassen oder die Oberfläche neugestalten. So könnten für die Desktop-Version neue Funktionen und Inhalte hinzugefügt werden, um den Platz besser auszufüllen. So ist es auch möglich, auf andere Ansprüche bei dieser Version einzugehen. Während bei der mobilen Version die reine Informationsvermittlung im Vordergrund steht, ist es möglich, auf der Desktop-Version diese und mögliche weitere Inhalte besser zu visualisieren.

Nach der Optimierung für Windowsstandards der mobilen Version, kann darauf aufbauend die Anpassung an andere Windows 10 Geräte erfolgen. Diese Umsetzung wird aber nicht im Rahmen dieser Bachelorarbeit umgesetzt und wäre Teil fortführender Projekte.

4. Anpassung der Halle-App

In diesem Abschnitt werden die nötigen Änderungen an der Halle-App erläutert.

4.1. Strukturanpassungen

Navigation:

Um die allgemeine Navigation zu verbessern, wurden die zwei obersten Navigationsebenen in eine einzelne mit einer Pivot-Navigation umgewandelt. Dadurch wird die App wesentlich übersichtlicher, da nun die meisten Seiten mit einem Klick erreichbar sind. Man muss nicht mehr durch zwei Ebenen navigieren. Mit der Pivot-Navigation kann man nun alle ehemals auf der obersten Navigationsebene vorhandenen Buttons durch nach rechts oder nach links wischen erreichen.

An den grundlegenden Kacheln auf den ehemals zwei Hauptnavigationsseiten wurden keine Änderungen vorgenommen. Diese passen sich aktuell schon an die Bildschirmgröße an. Für mobile Geräte muss hier nicht optimiert werden.

Im Rahmen des Responsive Design wären für größere Bildschirme anderer Gerätefamilien (wie z.B. Desktopmonitore etc.) noch weitere Strukturanpassungen notwendig.

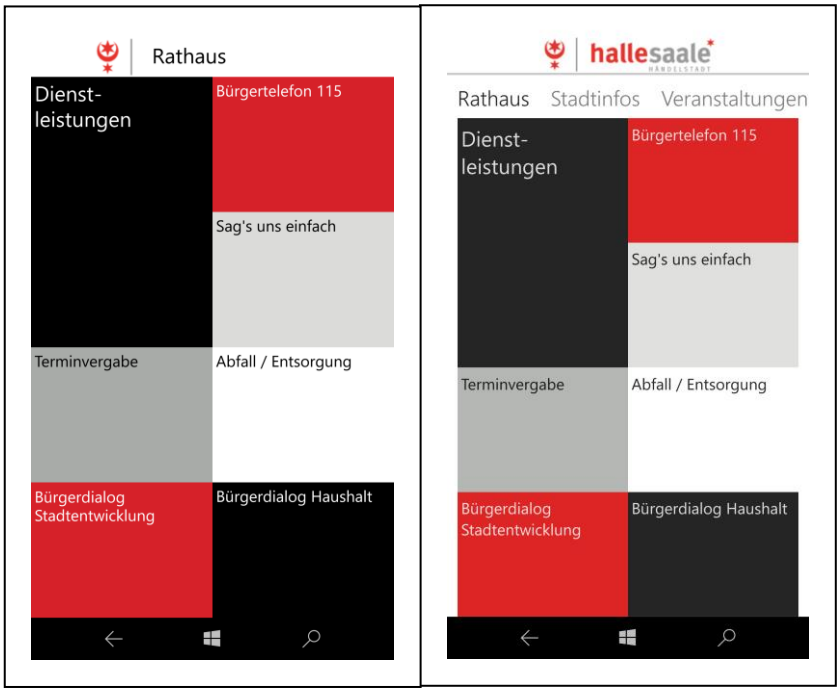

Abbildung 16 Menüanpassungen (vorher – nachher)

Inhalt:

Die Inhaltsseiten, die zu externen Internetseiten führen, wurden nicht verändert und ihre Funktionalität wurde beibehalten. Es gab keinen Handlungsbedarf.

Die internen Inhaltsseiten aus der Datenbank wurden ebenfalls in gleicher Weise übernommen. Da diese Seiten gut von der Navigation getrennt sind und man dadurch nicht beim Lesen gestört wird, entsprechen sie den Merkmalen einer konsumorientierten App. Inhaltlich mussten keine Veränderungen vorgenommen werden, da alle Informationen beibehalten werden sollen.

4.2. Designanpassungen

Farbe:

Die allgemeine Farbgestaltung entspricht den Richtlinien von UWP-Apps. Die von Microsoft verwendeten Akzentfarben sind in der App nicht vorhanden.

Wie bereits erwähnt, entspricht das helle Design der vorgesehenen Verwendung der App. So mussten keine Änderungen vorgenommen werden.

Zusätzlich wurden die Hintergründe der Überschriften der einzelnen Seiten farblich angepasst, um visuelle Verknüpfungen zwischen den Seiten zu schaffen und damit die Animationen zu unterstützen. Es erleichtert dem Nutzer das Nachvollziehen von Verbindungen zwischen den Seiten.

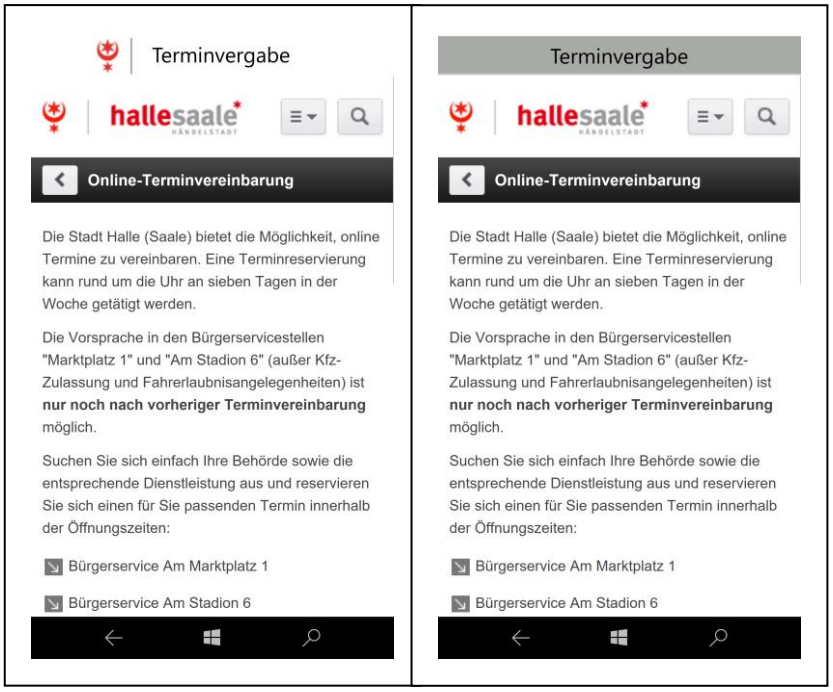

Abbildung 17 Menüanpassungen Untermenü (vorher – nachher)

Typografie:

Innerhalb der gesamten App wurde die Schriftart von Source Sans pro auf SegoeUI geändert.

Anderweitig sind keine Änderungen notwendig, da die Typografie-Richtlinien bereits standardmäßig bei Visual Studio eingestellt sind.

Symbole:

Es mussten keine Symbole optimiert werden.

Bewegungen:

Für eine dynamische und ansprechende Navigation wurden verbundene Animationen eingefügt. Durch diese wird eine visuelle Verknüpfung zwischen den Seiten geschaffen. Nachdem auf einen Button geklickt wurde, scheint dieser, sich in die Überschrift der neuen Ansicht zu transformieren. Dabei springt er von seiner aktuellen Position zu seiner neuen.

Zwischen den tieferen Navigationsebenen wurden einfache Seitenübergänge für eine bessere Dynamik hinzugefügt.

5. Ergebnis

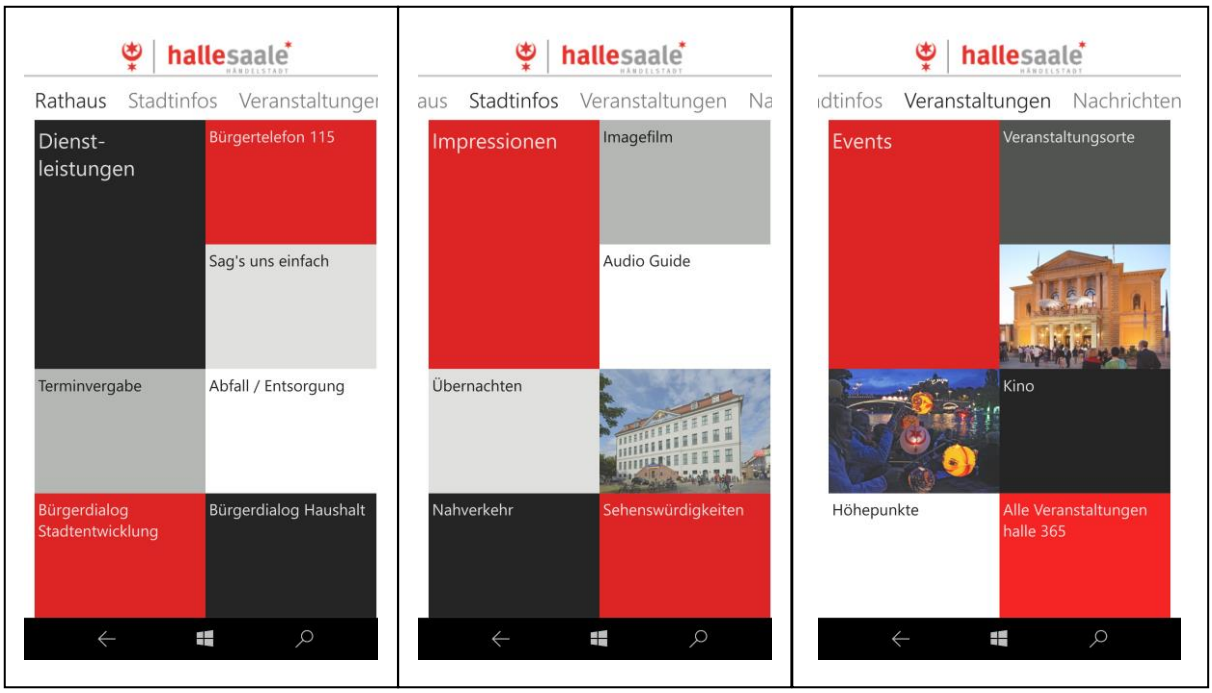

Abbildung 18 Hauptmenü UWP App

In den drei Bildern sieht man nun die wichtigsten Ausschnitte der obersten Ebene der App. Die ehemaligen Untermenüs „Rathaus“, „Stadtinfos“, „Veranstaltungen“ und ebenfalls „Nachrichten“ und „Baustellenkalender“ sind nun auf einer Ebenen untergebracht. Die Funktionalität hat sich ansonsten nicht geändert.

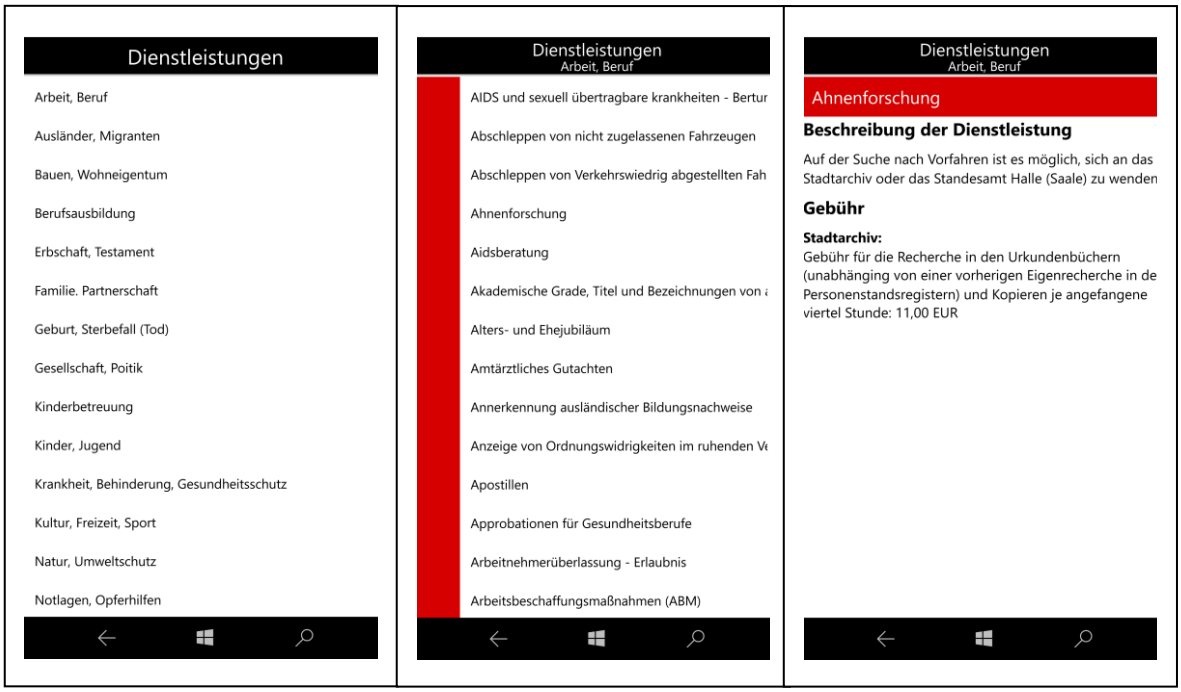

Abbildung 19 Untermenü UWP App (Inhaltsmenüs)

Die Funktionen der unteren Ebenen sind nur teilweise im Prototyp vorhanden. Aber um die Anpassungen an der Halle-App darzustellen, wurden diese, wie in Abbildung 19 zu sehen, angedeutet. Wie bereits erwähnt, wurden die Überschriften farblich angepasst und Animationen zwischen den Ebenen Wechseln eingefügt. So wurde eine bessere optische Verbindung zwischen den Ebenen geschaffen.

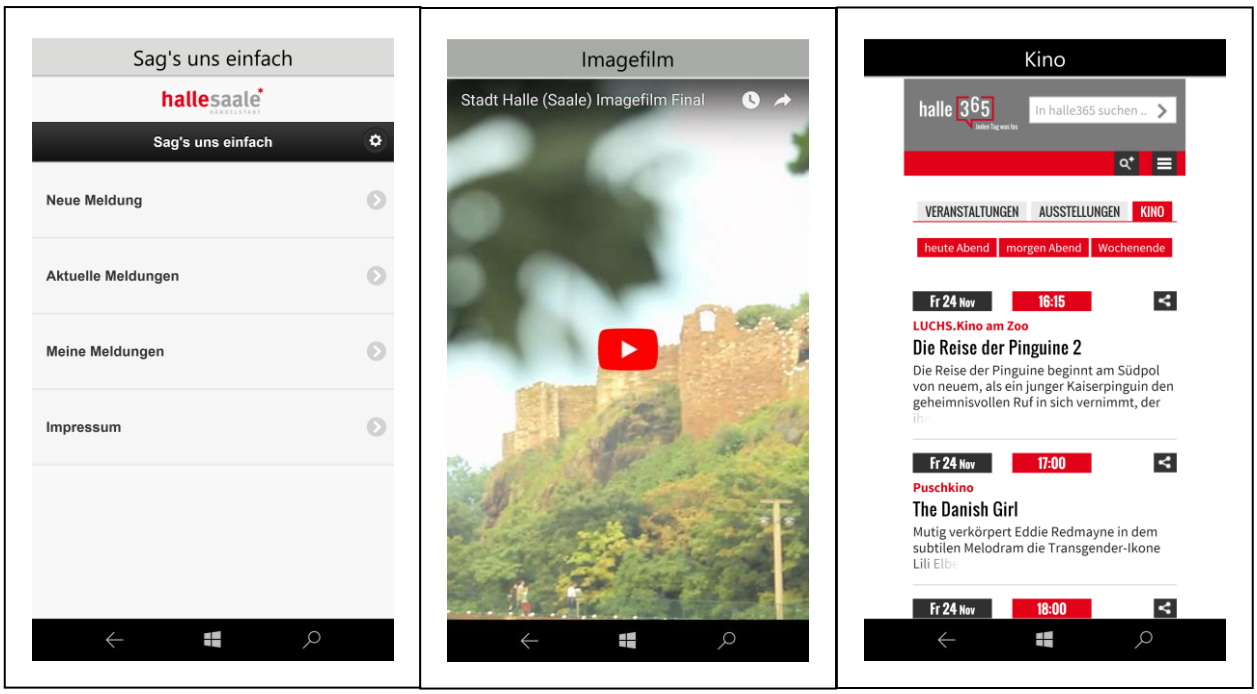

Abbildung 20 Untermenüanpassungen UWP App (Webeinbindungen)

An der Funktionalität der extern eingebundenen Seiten wurde nichts geändert. Auch die Möglichkeiten einer Optimierung sind hier nur gering. Wie auf den anderen unteren Ebenen wurden, hier ebenfalls Überschriften farblich angepasst und Animationen eingefügt.

6. Zusammenfassung und Ausblick

In der Halle-App wurden mehrere Veränderungen vorgenommen, um diese an Microsoft Standards anzupassen. Die Anpassungen haben hauptsächlich die Navigation verbessert. Außerdem wurden bessere optische Verbindungen zwischen den einzelnen Seiten durch Animationen und Farben geschaffen.

Darauf aufbauend können nun Änderungen hinsichtlich des Responsive Design und der verschiedenen Plattformen vorgenommen werden. In 3.4. Responsive Design wurde bereits darauf eingegangen, wie dies gestalterisch erfolgen könnte. Technisch müssten jetzt verschiedene Versionen für die verschiedenen Geräteklassen angefertigt werden, die alle universelle APIs verwenden. Der Code für die Funktionen würde bestehen bleiben, aber hinsichtlich des Designs sind Änderungen, wie zum Beispiel die reaktiven Designtechniken, nötig. Abschließend können nun alle Versionen bei Microsoft hochgeladen werden und tauchen als eine App in jedem Store der jeweiligen Geräteklasse auf.

Microsoft hat mit der Entwicklung der UWP App neue Möglichkeiten für Entwickler geschaffen. So kann nun mit einer App, durch die Entwicklung für mehrere Plattformen eine viel größere Zielgruppe angesprochen werden. Mit der neuen Technologie sind auch viele Anpassungs- und Optimierungsmöglichkeiten hinzugekommen. Es sollen immer mehr Apps für Windows 10 optimiert werden [Semmler].

Im Dezember 2017 wurde berichtet, dass Apple jetzt an einer ähnlichen Technologie entwickelt, die auf dem gleichen Prinzip wie eine Universelle Windows Plattform funktioniert. Es soll auch möglich sein eine App, auf verschiedenen Apple Geräten zu nutzen [Gurman].

Daran ist zu erkennen, dass auch andere Unternehmen den Nutzen und die Möglichkeiten einer universellen App-Plattform erkannt haben und dies nun kopieren wollen.

Es ist zu beobachten, dass die Hersteller von App-Entwicklerwerkzeugen die Technologie der universellen App-Plattform ausweiten. So ergeben sich für App-Entwickler weitere Möglichkeiten, den einmal erzeugten Code effizient an neue Geräteklassen anzupassen.

7. Literaturverzeichnis

[Microsoft 1] Einführung in die Universelle Windows-Plattform. https://docs.microsoft.com/de-de/windows/uwp/get-started/universal-application-platform-guide [03.01.2018].

[Microsoft 2] Screen sizes and break points for responsive design. https://docs.microsoft.com/de-de/windows/uwp/design/layout/screen-sizes-and-breakpoints-for-responsive-design [03.01.2018].

[Microsoft 3] Was ist eine Universelle Windows-Plattform (UWP)-App?. https://docs.microsoft.com/de-de/windows/uwp/get-started/whats-a-uwp [03.01.2018].

[Microsoft 4] Introduction to UWP app design. https://docs.microsoft.com/de-de/windows/uwp/design/basics/design-and-ui-intro [03.01.2018].

[Microsoft 5] Responsive design techniquess. https://docs.microsoft.com/de-de/windows/uwp/design/layout/responsive-design [03.01.2018].

[Microsoft 6] Navigation design basics for UWP apps. https://docs.microsoft.com/de-de/windows/uwp/design/basics/navigation-basics [27.10.2017].

[Microsoft 7] Command design basics for UWP apps. https://docs.microsoft.com/de-de/windows/uwp/design/basics/commanding-basics [27.10.2017].

[Microsoft 8] Content design basics for UWP apps. https://docs.microsoft.com/de-de/windows/uwp/design/basics/content-basics [27.10.2017].

[Microsoft 9] Color. https://docs.microsoft.com/de-de/windows/uwp/design/style/color [03.01.2018].

[Microsoft 10] Typography. https://docs.microsoft.com/de-de/windows/uwp/design/style/typography [03.01.2018].

[Microsoft 11] Motion for UWP apps. https://docs.microsoft.com/de-de/windows/uwp/design/motion/index [03.01.2018].

[Microsoft 12] Acrylic material. https://docs.microsoft.com/de-de/windows/uwp/design/style/acrylic [27.10.2017].

[Microsoft 13] Acrylic material. https://docs.microsoft.com/de-de/windows/uwp/design/style/acrylic [03.01.2018].

[Microsoft 14] Parallax. https://docs.microsoft.com/de-de/windows/uwp/design/motion/parallax [03.01.2018].

[Wiki 1] Surface Hub. https://de.wikipedia.org/wiki/Surface_Hub [03.01.2018].

[Wiki 2] Microsoft HoloLens. https://de.wikipedia.org/wiki/Microsoft_HoloLens [03.01.2018].

[Wiki 3] Internet der Dinge. https://de.wikipedia.org/wiki/Internet_der_Dinge [03.01.2018].

[Allsopp] John Allsopp: A Dao of Web Design. http://alistapart.com/article/dao [03.01.2018].

[Gurman] Mark Gurman: Apple Plans Combined iPhone, iPad & Mac Apps to Create One User Experience. https://www.bloomberg.com/news/articles/2017-12-20/apple-is-said-to-have-plan-to-combine-iphone-ipad-and-mac-apps [03.01.2018].

[Rohles] Björn Rohles: Grundkurs Gutes Webdesign. Galileo Press, Bonn, 2013.

[Semmler] Jan Semler: App-Design. Alles zur Gestaltung, Usability und User Experience. Rheinwerk Verlag GmbH, Bonn, 2016.

8. Abbildungsverzeichnis

Alle hier nicht eigens nachgewiesenen Abbildungen stammen vom Autor.

Diese Arbeit wurde von mir selbständig verfasst und in gleicher oder ähnlicher Fassung noch nicht in einem anderen Studiengang als Prüfungsleistung vorgelegt. Ich habe keine anderen als die angegebenen Hilfsmittel und Quellen, einschließlich der angegebenen oder beschriebenen Software, verwendet.