Institut für Parallele und Verteilte Systeme

Abteilung Anwendersoftware

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Bachelorarbeit Nr. 121

Vom Bedarf zur fertigen App

Vincenzo Baldini

Studiengang:	Informatik
Prüfer:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer:	Dipl.-Inf. Eva Hoos
begonnen am:	08. April 2014
beendet am:	25. Oktober 2014
CR-Klassifikation:	D.1.0, D.1.5, D.2.2, D.2.1, D.2.3

Kurzfassung

In dieser Arbeit wird der Entwicklungsprozess für Windows 8 Apps betrachtet. Beginnend bei der Einführung in das Betriebssystem Windows 8 und weiterführend in die Entwicklungsumgebung und deren Feinheiten. Anschließend folgt die Erstellung eines Kriterienkatalogs, der einen holistischen Entwicklungsprozess für Windows 8 Apps unterstützt und während der Entwicklung begleitet. Neben dem Entwicklungsprozess wird auch das Deployment über den Windows Store und den internen Gebrauch als Branchen App betrachtet. Es folgt eine Evaluierung dieses Kriterienkatalogs anhand einer für diesen Zweck entwickelte App. Es wurde vorrangig die Entwickler Website von Microsoft als direkte Quelle verwendet, umso, Zweitmeinungen zu vermeiden. Die Evaluierung des Kriterienkatalogs anhand der entwickelten App zeigte, dass dieser als Grundlagen Checkliste und Leitfaden sehr gute Ergebnisse erzielt. Mit dieser Erkenntnis kann der Kriterienkatalog für Einsteiger in die Windows 8 App Programmierung effizient genutzt werden und unterstützt den Entwickler begleitend durch den Entwicklungsprozess von Windows 8 Apps.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung & Motivation – Vom Bedarf zur Fertigen APP

Mit nur einem Griff in die Hosentasche hält man praktisch die Welt in Händen. So geht es jedem der ein Smartphone besitzt. Entertainment, E-Mails checken oder einfach nur mit den Freunden chatten, durch mobiles Internet ist ein uneingeschränkter Zugriff auf so gut wie allen Dienste, die es in Form von Apps gibt, möglich.

Wenn man betrachtet, dass die Entwicklung von Smartphones bereits Mitte der Neunziger Jahre begonnen hat und das erste Multitouch-Device erstmals von Apple im Jahre 2007 vorgestellt wurde, erkennt man, wie schnell der Entwicklungs- und Vermarktungsprozess über wenige Jahre voranschritt.

Dabei lag der Focus zu Beginn primär auf der Entwicklung von Smartphones, die dem User die Möglichkeit boten, über einen Touchscreen mit dem Gerät zu interagieren. Dabei gewannen die Smartphones sehr schnell an Leistung, wodurch komplexere und aufwändigere Programme auf den Geräten ausgeführt werden konnten. Auch nahm die Qualität des Touchscreens über die Zeit stark zu und was damals ein resistives Display war, bei dem die Bedienung nicht ganz so reibungslos ging, ist heute ein Full-HD-Super-AMOLED-Screen der bereits einen Finger oder Eingabestift in der sich nur in der Nähe des Displays befindet erkennt.

Durch die ständige Verbesserung und weiter Entwicklung dieser High-End-Devices, legten Smartphones den Grundbaustein für Tablets, welche von dem User meist wegen der größeren Displays, genauso wie dem Touch Komfort und der dadurch resultierenden leichten Bedienung, genutzt werden. Diese genannten Gründe, führten zu einer Favorisierung von Tablets gegenüber Notebooks. Das kleinere Volumen der Geräte sowie das Fehlen von sperrigen Gadgets wie Tastatur oder Maus, bestärken die positiven Argumente für Tablets.

Da Tablets aber bereits in den späten 90er als eine Vorentwicklungsstufe von Laptops entwickelt wurden, dennoch nie Fuß fassen konnten, fanden diese erst durch die Vorar-

beit der Smartphones und das langsame herantasten des Users an Touch Devices ihren Platz in der Gesellschaft.

Mittlerweile lösen Smartphones und Tablets in vielen Nutzungsbereichen den herkömmlichen PC und Laptop ab. Und so haben Tablets wie das iPad, Google Nexus und Amazon Kindl bereits seit Mitte 2013 höhere Verkaufszahlen als Notebooks.

Bei diesem Stand der Technik stellt sich dem Verbraucher die Frage, nach dem passenden Betriebssystem für sein Gerät. Und wie in den meisten technischen Bereichen scheiden sich hier die Gemüter.

Vorrangig haben wir die beiden Spitzenreiter Apple, mit seinen iOS Geräten, und Android mit dem gleichnamigen Betriebssystem für Smartphones und Tablets. Wo Apple nur auf seine eigenen Geräte setzt, flutet Android den Markt mittels verschiedener Smartphone-Hersteller die sich die Android-Software personalisieren dürfen.

Doch seit Ende des Jahres 2012 bahnt sich ein neuer Konkurrent seinen Weg in die Welt der Tablets und Smartphones. Mit seinem neuen Betriebssystem Windows 8 und der damit verbundenen Spezialisierung auf Tablets und Smartphones, bestreitet Microsoft seinen Weg zurück in diese Techniknische.

Laut Microsoft nutzen 1,3 Milliarden User Windows-Betriebssysteme auf Ihren Geräten (Microsoft 2014a). Dabei gibt das Unternehmen allerdings nur die Gesamtzahl der Downloads auf Ihrer Webseite preis und nicht die der tatsächlichen Nutzer. Dies lässt sich auf einen Anteil von ca. 10-12% schätzen, sofern man Windows 8.0 und 8.1 im Gesamten betrachtet. Daraus folgt, dass Windows 8 seit seinem Release im Oktober 2012 täglich von ca. 130 Millionen Menschen genutzt wird.

Dabei gibt es im allgemeinen Windows-App-Store rund 140.000 Apps, die bereits 250 Millionen Downloads zählen (Microsoft 2014a). Wenn man diese Zahl auf die User verrechnet, ergibt das einen Schnitt von ca. 2,5 Apps pro User.
Dies hört sich verhältnismäßig wenig an, wenn man betrachtet, wie hoch die Anzahl an Apps ist, die ein Mac-User im Schnitt downloaden muss, um ein einigermaßen an sich selbst angepasstes System zu erreichen, mit dem der User arbeiten oder Entertainment genießen kann. Die Erklärung hierfür ist im Prinzip doch sehr einfach, denn im Gegensatz zu anderen Betriebssystem hat Windows sehr viel vorinstalliert. Nach dem Prinzip

„einstöpseln und loslegen“ muss der User nur sehr wenig ergänzen um das System anzupassen. Auch der Windows Phone-Store beinhaltet 240.000 Apps und Games(Microsoft 2014a), die insgesamt 2 Milliarden Mal heruntergeladen wurden.

Genauso wie im privaten Gebrauch, nehmen durch die ständige Innovation der Geräte die Anforderungen in Unternehmen immer mehr zu. Um im Enterprise-Bereich möglichst effizient Apps oder Programme zu entwickeln, stellt sich neben der Frage, welches Betriebssystem zu nutzen ist, sofort die Frage, welches Gerät dem Nutzen und Zweck der App am besten gerecht wird. Um diese Fragen möglichst gut zu beantworten, muss dabei genau der Zweck und der jeweilige Nutzer betrachtet werden, denn es ist nicht sehr sinnvoll einen IT-Consultant mit einem PC auszurüsten, der Ihm die Mobilität nimmt. Genauso wenig effizient wäre die Entscheidung eines Technischen Zeichners seine CAD Zeichnungen auf einem Tablet zu realisieren, welches dem Rechenaufwand nicht standhalten kann.

In dieser Arbeit wird das Betriebssystem Windows 8.1 genauer untersucht und beschrieben. Microsoft stellt klar seine Marktdominanz im Bereich Desktop PCs in den Raum und die hohen Nutzerzahlen von Windows 7, die seit dem Absatz von Windows XP stetig wachsen (NET). Der Umschwung in Unternehmen von Windows XP auf das Betriebssystem Windows 7 wird als Sprungbrett für den Gebrauch von Windows 8 genutzt. Als weiteres Argument, dass für Unternehmen sehr wichtig sein wird, ist die Aussicht auf eine Multi-Device Programmierung. So wird eine App im neuen Windows-Modern-UI Design über den Microsoft Shop für PCs, Laptop, Tablets und mit der richtigen Programmierung sogar für Smartphones bereitgestellt. Im Gegensatz zu Apple oder Android hat Microsoft mit Windows 8 die Möglichkeit geboten, Device-übergreifend zu programmieren.

Trotzdem ist es manchmal wenig sinnvoll eine App für mehrere Geräte zu programmieren. Wie das oben aufgeführte Beispiel bereits zeigt, müssen dabei bestimmte Kriterien beachtet werden. In dieser Arbeit werden wir zu Beginn die Voraussetzungen und Besonderheiten für und in der Entwicklung von Windows 8 betrachten. Des Weiteren betrachten wir das Windows-Modern-User-Interface und stellen sicher, dass wir besondere Rücksicht auf den Nutzer nehmen. Dieses Konzept des User-Centered-Design wird im späteren Verlauf eine wichtige Rolle in der Erstellung unseres Kriterienkataloges spielen.

Der in dieser Arbeit erstellte Kriterienkatalog hat die Aufgabe durch vordefinierte Kriterien den Entwicklungsprozess einer App zu beschleunigen und zu vereinfach. Dabei bauen die Kriterien sukzessive aufeinander auf und können beim Erstellen einer App als Checkliste verwendet werden. Im weiteren Verlauf wird der Kriterienkatalog in Form einer Checkliste verwendet um die zu implementierende Windows 8 App mit dem Namen „DestinAPP“ (Decision Support to Identify Beneficial Apps) zu erstellen und zu bewerten. Die DestinApp beruht auf den Errungenschaften des „Analysis Framework to Identify Value-added App Usage Scenarios“ (HGKM), welches im späteren Verlauf genauer erläutert wird und implementiert dieses Framework. Anschließend folgt eine kritische Betrachtung des Kriterienkatalogs und eine Evaluierung der Kriterien anhand der erstellten DestinAPP.

2 Einführung in Windows 8

Mit Microsofts neuem Betriebssystem Windows 8 eröffnet sich ein neues Spektrum der Programmentwicklung. Im folgenden Teil dieser Arbeit werden Ihnen die Grundvoraussetzung zur Entwicklung in und für Windows 8 vorgestellt. Im späteren Verlauf werden auf die Besonderheiten im Deployment sowohl als Branchen App interhalb Unternehmen, als auch für den öffentlichen Gebrauch und deren Vermarktung eingegangen.

2.1 Beobachtungen und erste Nutzung

2.1.1 Bootvorgang

Die Bootzeit von Windows 8 hat sich stark verkürzt. Dies wird mittels Fastboot von Windows 8 und bei neuen Rechnern aus einer Kombination zwischen der Fastboot-Technologie und dem UEFI BIOS „Unified Extensible Firmaware Interface“, welches nur für 64bit Systeme zur Verfügung steht, realisiert. Dabei besteht der Fastboot-Vorgang aus einer Kombination aus Starten aus dem heruntergefahrenen System und dem Ruhezustand (PCW). Windows 8 fährt, sofern Fastboot aktiviert ist, nicht mehr komplett

herunter, sondern wechselt in den Ruhezustand, bei dem aber nicht der komplette Systemzustand gespeichert wird, der auch die User-Session beinhaltet, sondern lediglich den Kernelzustand. Dies hat den Effekt, dass eine neue Windows-Session beim Start des Systems gewährleistet ist. Microsoft bietet bei einem Multiboot-System die Möglichkeit, die Fastboot-Option abzuschalten, da durch das Wechseln zwischen mehreren Betriebssystemen die gespeicherte Session sowieso bei jedem Wechsel gelöscht wird und der Bootvorgang somit komplett neu gestartet wird.

2.1.2 Startseite

Nach dem Bootvorgang befindet man sich auf dem neuen Homescreen, oder auch Startseite genannt, der sich absolut vom gewohnten Design, wie es aus Windows 7 und früheren Versionen bekannt ist, unterscheidet. Das neue Modern-UI Design von Microsoft (SB13) bietet damit die Kachelansicht auf der Startseite und zeigt alle installierten favorisierten Apps an und ist, was Position und Größe der Kachel der Apps betrifft, sofern der Entwickler der App das vorgesehen hat, vollständig anpassbar. Des Weiteren kann der User direkt über die Kacheloptionen eine App deinstallieren oder mehrere Apps in Gruppen zusammenfassen, denen man optional einen Gruppennamen geben kann. Um auf alle installierten Programme zugreifen zu können, bietet die Startseite über einen suboptimal positionierten und unbeschrifteten Pfeil nach unten die Option, zu einer Liste aller installierten Programme zu gelangen und diese von dort zu starten, der Startseite hinzuzufügen oder deinstallieren zu können.

2.1.3 Desktop

Die Nutzung der Startseite zeigt, dass man nicht, wie man es aus dem bisherigen normalen Gebrauch gewohnt ist, nach unten Scrollen kann, sondern nur noch die horizontale Ansicht benutzt. Was für Touchgeräte kein Problem darstellt, wirkt aber bei einem Gerät das mittels Maus oder ähnlichen Steuergeräten genutzt wird, doch etwas umständlich. Weiter bietet Windows 8 über diese Kachelansicht, dem User die Möglichkeit auf eine Desktopversion zu gelangen, die dem Standard-Desktop der Windows Vorgängerversionen ähnelt. Da die Navigation der Desktopansicht über Ordnerstrukturen und Menu-Optionen in der Ordneransicht erfolgt, scheint die Navigation in bestimmte Be-

reiche, wie beispielsweise die Systemsteuerung, für ungeübte Windows-Nutzer erschwert. Trotzdem stellt der Desktop alle Funktionalitäten, wie das Installieren und Starten von Desktopprogrammen und weiteren Funktionen, die man aus den Windows 8 Vorgängerversionen gewohnt ist, zur Verfügung. Obwohl der Desktop selbst streng genommen nur eine Kachel-App ist und in Windows 8 wie eine solche behandelt wird.

2.1.4 Apps und Programme

Windows 8 ermöglicht es, sowohl native Programme früherer Windowsversionen als auch besagte Kachel-Apps zu nutzen. Dabei differenziert Windows 8 ab einem gewissen Punkt sehr stark bei deren Nutzung. Native Programme können zwar der Startseite als Kachel hinzugefügt werden, können aber zum Beispiel keine Live-Ticker Informationen als Kachel anzeigen und weitere Features wie Windows-Toast und Animationen nutzen. Der Start eines nativen Desktopprogramms erfolgt wie bei Windows 8-Apps über die Startseite oder als Verknüpfung auf dem Desktop der Desktopansicht. Das gestartete Desktopprogramm wird in der Desktopansicht dargestellt und bedient. Im Gegensatz dazu werden Windows-Store Apps, wie die Desktopansicht selbst, gestartet und als eigenständiges Programm betrachtet. Demnach ist ein Wechseln zwischen Desktopansicht und Windows 8 App über das linke Charm-Menu möglich. Im Falle, dass mehrere Apps genutzt werden, kann man zwischen diesen über das besagte Charm-Menu navigieren, sie auf dem Bildschirm als Split Screen einstellen oder schließen.

Der Start einer Windows-Store App erfolgt über das Kachelsystem und wird nicht in der Desktopansicht angezeigt. Betrachtet man eine für Windows 8 erstellte App, bemerkt man das gleiche Designkonzept wie schon auf der Startseite zu sehen war – der Scroll Vorgang geht nur in der Horizontale. Dabei strukturiert sich eine App über die von Microsoft spezifizierten User Experience Guidelines.

2.2 Voraussetzungen für die Entwicklung in Windows 8

2.2.1 Betriebssystem

Die Entwicklung von Windows Store- und Windows Phone-Apps ist ausschließlich mit Windows 8, beziehungsweise nur mit der neuesten Version Windows 8.1, möglich. Damit erzwingt Microsoft ein in sich geschlossenes System, welches mit der Strategie von Apple vergleichbar ist, Software nur mit den eigenen Geräten produzieren zu können.

Eine Möglichkeit um mittels anderer Betriebssysteme oder Vorgängerversionen eine Windows App zu entwickeln, wäre die Nutzung einer virtuellen Maschine, in der man eine Version von Windows 8 nutzt. Dieses Verfahren ist aufgrund der Anforderung hoher Rechnerressourcen nur eine langsame und deshalb suboptimale Lösung. Eine weitere Möglichkeit wäre das Einrichten mehrerer Partitionen auf einem Gerät. Dabei wäre sicherzustellen, dass für das gewählte Gerät, sofern dies ein älteres Modell ist, die Treiber der entsprechenden Hardware zur Verfügung stehen.

Des Weiteren besteht Microsoft darauf, dass ein Update auf Windows 8.1 ausgeführt wird, welches, wenn man bereits im Besitz von Windows 8 ist, kostenlos über den Windows-Store bereitgestellt wird und einige neue Ansätze, auch bezüglich der Funktionalität im Bootvorgang, so wie Neuerungen in Windows RT für Tablet-Computer beinhaltet. Führt man das Update nicht durch, verweigert Microsoft die Installation der Entwicklungsumgebung Visual Studio 2013, da Windows 8.1 als einziges unterstütztes Betriebssystem vorausgesetzt wird. Somit wird dem Entwickler eine klare Richtlinie vorgegeben, aber dafür wird gleichzeitig ein gewisser Standard bezüglich der Konsistenz, Verlässlichkeit und Sicherheit in der Entwicklung von Windows Apps gewährleistet.

2.2.2 Hardwarevoraussetzungen

Es treffen die gleichen Mindestvoraussetzungen an das System zu wie für die Vorgängerversion Windows 7:

- 1GHz Prozessor
- 1 GB RAM (für 32bit) und 2 GB (für 64bit) Systeme.
- Mindestens 16GB Festplattenspeicher nur für die Installation von Windows benötigt.
- Eine Grafikkarte mit DirectX 9 und WDDM-Treiber

Damit macht es Microsoft möglich PCs und Notebooks, die im Moment noch mit Windows 7 laufen, ohne Hardwareersatz, auf Windows 8.1 umzurüsten. Eine Strategie zugunsten Microsofts, um die hohen Nutzerzahlen von Windows 7 abzugreifen.

Außerdem setzt Windows 8/8.1 eine Internetverbindung und eine Bildschirmauflösung von mindestens 1024 x 768 Pixel voraus, um überhaupt Apps aus dem Store herunterladen und installieren zu können. Sofern diese Voraussetzungen erfüllt sind und man den Store nutzen möchte, oder jede andere App die von Microsoft schon bereits nach der Installation vom Betriebssystem bereit gestellt wird, wird man dazu aufgefordert, ein Benutzerkonto anzulegen, das dazu genutzt wird, einen Zugang zum Store und jeder anderen vorinstallierten App zu bekommen.

2.3 Die Entwicklungsumgebung

Um Apps für den Windows Store oder für das Windows Phone zu entwickeln, bietet Microsoft die Entwicklungsumgebung Visual Studio an. Dabei sieht Microsoft keine weitere Entwicklungsumgebung vor. Dieses Kapitel beinhaltet die Wesentlichen Aspekte zur Nutzung von Visual Studio und dessen Anforderungen.

2.3.1 Microsoft Visual Studio

Microsoft Visual Studio 2013 beinhaltet eine ausgewählte Sammlung von Tools zur Erstellung und Codierung sowie zum Debuggen, Lokalisieren, Packen und Bereitstellen der App für den Windows Store oder Windows Phone-Store. Dabei bezieht sich das neue Visual Studio 2013 ausschließlich auf Windows 8.1 und kann in Windows 8 weder genutzt noch installiert werden.

Die Verwendung auf einem anderen Betriebssystem als Windows 7 ist nicht möglich. Sollte es aber der Fall sein, dass die Entwicklung für Windows 8 bestimmt ist, so besteht die Möglichkeit dies mit Visual Studio 2012 zu tun. Dabei ist noch zu bedenken, dass die Entwicklung von Windows Phone-Store Apps ausschließlich mit Visual Studio 2013 und dem Update 2 möglich ist. Dazu wird zusätzlich das Windows Phone SDK 8.0 benötigt welches installiert werden muss.

Ein Update auf Windows 8.1 ist unumgänglich, und somit ist es nur noch möglich mit Visual Studio 2013 als Entwicklungsumgebung zu arbeiten. Durch dieses konsequente Verhalten gewährleistet Microsoft eine hohe Konsistenz der Software und Kompatibilität der Versionen untereinander.

Hat man bereits im early state mit der Entwicklung von Windows 8 begonnen und bereits Projekte mit Visual Studio 2012 erstellt, so kann die Kompatibilität der höheren Version nutzen, um ältere Projekte von Windows 8 zu warten. Dies funktioniert solange wie keine neuen Funktionalitäten, die nur Visual Studio 2013 anbietet, hinzufügt. Ab dem Zeitpunkt, an dem Funktionen höherer Versionen hinzugefügt wurden, können die in Visual Studio 2013 geänderten Projekte nicht mehr in Visual Studio 2012 geöffnet werden.

Für den Gebrauch von Visual Studio 2013 ist eine Entwicklerlizenz nötig. Die Menge an Lizenzen ist unbegrenzt und kostenlos. Eine Entwicklerlizenz ist ausschließlich zum Erstellen einer App in Visual Studio nötig, ohne eine Entwicklerlizenz kann Visual Studio für Windows nicht genutzt werden. Des Weiteren muss eine Entwicklerlizenz nach 30 Tagen erneuert werden, was im Prinzip einer einfachen Bestätigung gleich kommt, dass diese Lizenz noch genutzt wird. Erstellte Apps können mit einer Entwicklerlizenz auf

dem Entwicklergerät oder über eine Domäne (Sideloading, mehr dazu im Kapitel 2.6 Deployment) auch auf anderen Geräten installiert und getestet werden.

Besteht Interesse an dieser Stelle Apps für den Windows Phone-Store zu entwickeln, ist es wichtig, sofern direkt auf einen Smartphone gedebugged werden soll, alle nötigen Geräte zu registrieren.

2.4 Programmiersprachen

Microsoft bietet eine Reihe an grundlegenden Komponenten und Optionen für die Entwicklung von Windows Store-Apps an. Ihr Zusammenspiel beginnt bereits bei der Wahl der Programmiersprache. Wo iOS und Android ausschließlich ihre eigenen Entwicklungssprachen verwenden, bietet Microsoft, wie auch in Abbildung 1 zu sehen ist, für die Entwicklung folgende Programmiersprachen an:

- JavaScript und HTML5.
- C# oder VisualBasic, welche beide in Kombination mit XAML (Extensible Application Markup Language) genutzt werden.
- C++ mit XAML oder DirectX.

Ein Entwickler kann sich unter diesen Programmiersprachen seine favorisierte aussuchen und für die Entwicklung nutzen. Das hat den Vorteil, dass Entwickler keine Zeit erlernen neuer Programmiersprachen verschwenden müssen, sondern sofort mit der Programmierung von Windows Store-Apps beginnen können. Allerdings ist hierbei zu beachten, dass jede Sprache ihre Feinheiten besitzt. Im direkten Vergleich wäre die Wahl für JavaScript/HTML5 als für kleine Projekte, welche nur zur Darstellung von Informationen dienen äußerst effizient, da diese schnell und simpel realisiert werden können. Aufwändige Projekte, welche Objektorientiert gelöst werden müssen, würden in die Kategorie C# oder VisualBasic fallen. Für eine Spieleentwicklung oder Rechneraufwändige Programme oder solche bei denen der Programmiere den Speicherbedarf kontrollieren muss, käme C++ mit DirectX oder XAML in Frage.

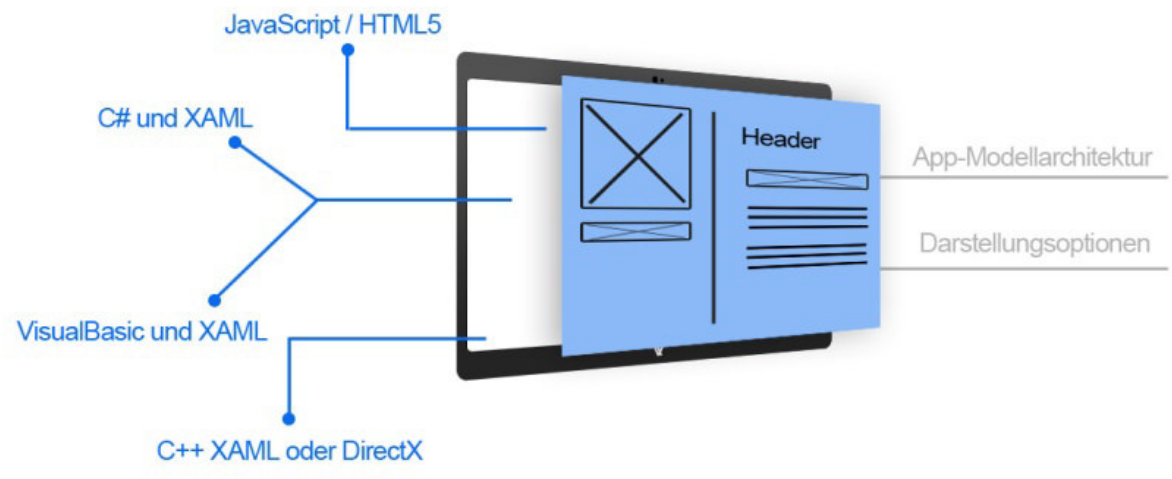

Abbildung 1 Mögliche Programmiersprachen für Windows 8 - Apps.

Microsoft bietet zu jeder Programmiersprache eine so genannte App-Modellarchitektur. Diese beinhaltet alle nötigten Daten einer App, wie zum Beispiel den Einstiegspunkt der Anwendung, welcher vom System aufgerufen wird, um den Code der App auszuführen oder auch verschiedene Designmuster. Im App-Modell bereitgestellte Designmuster, definieren das Erscheinungsbild der App. Dabei stehen einem Entwickler drei verschiedene Technologien zur Auswahl und zur freien Verfügung. Wie in Abbildung 1 zu sehen ist, ist an jede Programmiersprache eine Designsprache gekoppelt. Wird beispielsweise JavaScript als Sprache genutzt, so wird das Design mit HTML5 beschrieben. C#, VisualBasic und C++ nutzen XAML als Designsprache, wobei man sich bei C++ je nach Anwendungsfall auch für DirectX entscheiden kann.

„Extensible Application Markup Language“, mit XAML abgekürzt, ist eine von Microsoft bereitgestellte, allgemeine Beschreibungssprache, welche zur Oberflächengestaltung von Anwendungen genutzt wird. XAML basiert sehr stark auf der XML Schreibweise und stellt grafische Elemente, wie Buttons, Farbmuster, Fenster und vieles mehr dar, auch wird das Grid-System damit beschrieben und die Elemente auf einer Page positioniert.

Die Wahl der Programmiersprache hängt bei der Entwicklung für Windows Store-Apps nur von den Fähigkeiten des Entwicklers ab, es ist also nicht nötig eine zusätzliche Sprache zu lernen sofern der Entwickler bereits eine der vier genannten Programmiersprachen beherrscht. Dabei ist zu beachten, dass keine Vereinheitlichung der Sprachen durch Microsoft stattfindet; jede Sprache behält ihre eigene Syntax.

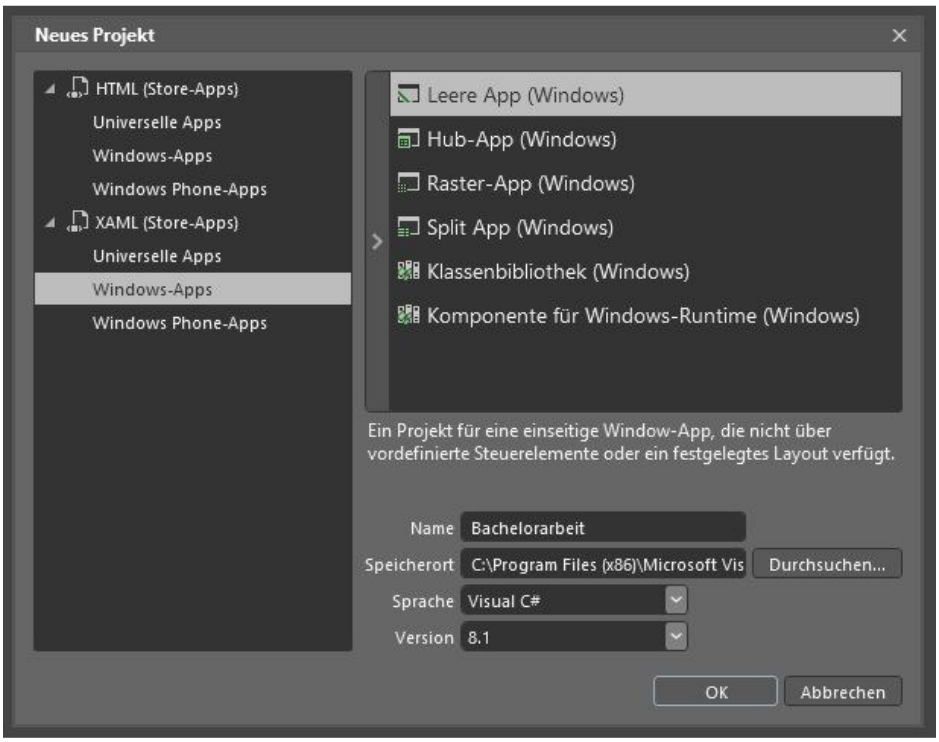

Abbildung 2 - Auswahl zur Projekterstellung in Visual Studio 2013.

2.5 Microsoft Modern – UI – Design

Die Struktur von Windows Apps unterscheidet sich grundlegend von bisherigen Desktop-Apps. Die ersten Unterschiede liegen bereits im Startscreen von Windows 8. Wo früher Desktopsymbole waren, sind jetzt interaktive Kacheln zu sehen über die, mit einem Klick oder einer Touch-Geste auf die jeweilige App zugreifen kann.

Nach dem Starten einer App öffnet sich die App-Startseite, dies ist die der Startpunkt einer jeden App und bildet die Grundlage für das entwickelte User Interface. Darin enthalten sind, alle Inhalte und Kontrollmöglichkeiten. Weiter dient die App-Startseite einem Entwickler als Basis, um dort UI Elemente anzuzeigen und zu strukturieren. In der Hierarchie können auf die Startseite weitere App-Seiten folgen, welche über Buttons oder entsprechende Elemente erreicht werden können.

Bei der Erstellung sollte Wert darauf gelegt werden alle Elemente einer App-Seite auch innerhalb dieser zu platzieren. Es wird von Microsoft und dessen User Experience Guidelines empfohlen anstelle von selbst definierten Popups, die standardmäßig integrier-

ten Animationen von Windows zu nutzen, da diese für alle Geräte definiert sind und es somit zu keinen Verschiebungen im Bildbereich kommen kann.

2.5.1 Fluid Design

Die standartmäßig integrierten Animationen sind besonders vorteilhaft in Betracht dessen, dass User von Windows 8 die Möglichkeit haben verschiedenen Ansichten von Apps zu nutzen, wie zum Beispiel den Bildschirm zu teilen, um so zwei Apps gleichzeitig auf dem Display anzeigen zu lassen. Diese Bildbereiche, werden von Microsoft in drei Kategorien geboten, zum einen gibt es den Full-View, welches den ganzen Bildschirm mit einer App füllt, und zum anderen die Splitt-Views, welche den Bildschirmbereich einmal um die Hälfte teilt oder den Bildschirmbereich in ein Drittel und zwei Drittel für zwei Apps strukturiert.

Windows zielt damit auf ein responsive Design ab. Dabei handelt es sich um ein Designpattern bei dem der Programmierer sicherstellt, dass sich das Design an jede Bildschirmgröße selbstständig anpasst und es zu keinen ungewollten Verschiebungen der Inhalte kommt. Dies wird meistens durch die Fluid Eigenschaft der Inhaltselemente erreicht und passt sich so jeder Bildschirmgröße an, hierbei wird der Fokus darauf gelegt für Elemente und das Grid-System prozentuale Höhen- und Breitenangaben, diese werden über die Bildschirmgröße zu Ihrer eigentlichen Größe berechnet. So kann bei verschiedenen Bildschirmgrößen oder Bildbereichen das Layout sich selbständig an die passende Ansicht skalieren. Damit ist eine Landscape- oder Portraitansicht und die damit verbundene Rotation des Bildschirms miteingeschlossen. Sollte sich der Entwickler dazu entscheiden, seine App für mehrere Geräte mit verschiedenen Bildschirmgrößen und Ansichten bereitzustellen, definiert er dies über die Darstellungsoption, wobei Windows durch Benutzung der Fluidansicht den Hauptteil der Arbeit abnimmt.

Abbildung 3 - Abstraktes Beispiel einer Hub-Page (Microsoft - User experience guidelines).

2.5.2 Navigationshierarchie

Ein Wichtiger Punkt der Konzeptionierung ist zu erkennen wie Windows die Navigation aufbaut, um die gewünschten Elemente zu organisieren. Windows versucht die Navigation durch die App so einfach und intuitiv wie möglich zu gestalten, um dabei den Inhalt bestmöglich in den Fokus zu setzen, setzt Windows auf eine hierarchische Navigationsstruktur, wie in Abbildung 4 zu sehen ist. Den Startpunkt bildet hierbei eine Hub-Page, wie sie in Abbildung 3 zu sehen ist. Diese hierarchische Strukturierung einer App ermöglicht eine einfache Bedienung, wodurch ein einfaches Navigieren gewährleistet wird und die App auch mit vielen Inhalten und Unterpunkten noch effizient bedienbar ist. Die Grundidee dieser Strukturierung einer App liegt vorrangig in der Gliederung des Inhaltes in verschiedene Bereiche, wobei jeder Bereich tiefergehend in der Navigation mehr Informationsgehalt zu dem gewünschten Thema liefert. Im Folgenden ein einfacher Ablauf der hierarchischen Struktur:

- Die Hierarchie beginnt bei einer Hub-Page, welche den Startpunkt der App bildet und die Teaser der verschiedenen Sections darstellt, dabei steht jede Section für einen eigenen, bestimmten Themenbereich. Des Weiteren können allgemeinen Informationen oder Optionen auf der Hub-Page dargestellt werden.

- Nach der Auswahl einer Section gelangt man in ihren Sectionsbereich, in dem noch weitere Inhalte zu diesem Bereich angezeigt werden können. Wird ein bestimmter Inhalt gewählt, so gelangt man auf die Detailansicht, die das Dritte Level der Hierarchie bildet.

- Die Detailansicht gibt den gewünschten Inhalt wieder. Das Design der gewünschten Information ist abhängig von der Informationsdarstellung. Dies können Videos, Texte, Bilder, Graphen oder eine Kombination aus allen Darstellungsmöglichkeiten sein.

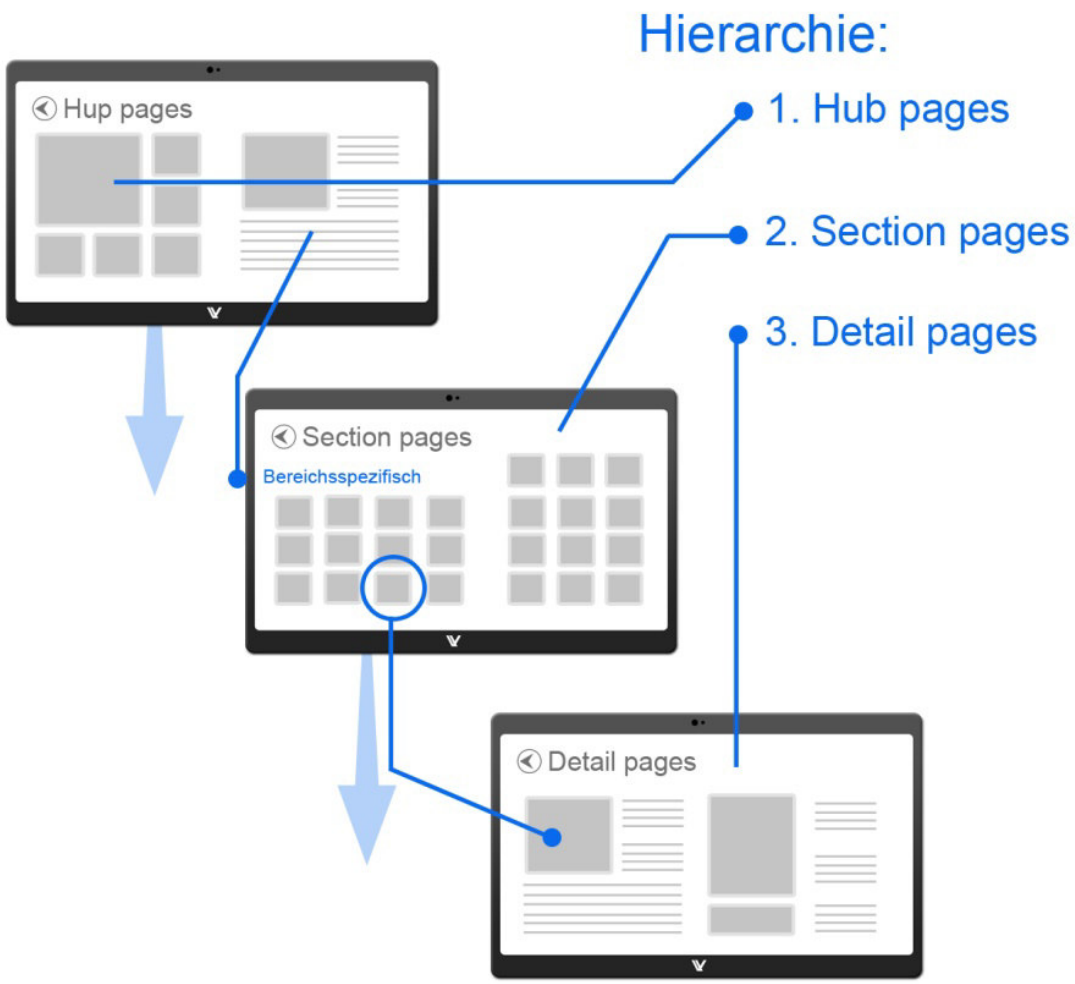

Abbildung 4 - Abstraktes Beispiel der Seitenhierarchie. Hub-Page, Section- und Detailpage (Microsoft - User experience guidelines).

Es wird nicht vorgeschrieben die Hub-, Section- und Detailpage in diesem Sinne zu verwende, allerdings sind sie zu diesem Zweck optimiert. Microsoft bietet zu diesem Thema eine weite Palette an neuen Informationen und Konventionen an, welche im Designguide von Microsoft zusammengefasst sind.

Letztendlich ist für einen Einstieg in eine einfache Entwicklung die Hierarchie und Menüführung das Ausschlaggebende. So kann bereits mit einem Template (siehe Abbildung 2) eine einfache Struktur erzeugt werden.

2.6 Deployment

Dieses Kapitel beschreibt das Deployment einer App. Eine App kann auf zwei verschiedene Weisen verteilt werden, zum einen durch die Bereitstellung der App im Windows Store oder, sofern es eine Phone App ist, im Windows Phone-Store und zum anderen als Branchen App für den internen Gebrauch in Unternehmen. Hier kommen unterschiedliche Vorgehensweisen zum Vorschein.

2.6.1 App im Store bereitstellen

Bei der Verteilung einer App im Windows Store oder Windows Phone-Store kommt es auf vier prinzipielle Aspekte an. Beginnend bei der Entscheidung für welchen Store die App bereit gestellt werden soll, gefolgt von den nötigen Informationen die für eine App bereit gestellt werden sollen und der spezifischen Wahl der Märkte und der dazu gehörenden Ländereinstellungen. Des Weiteren ist es nötig eine Altersbeschränkung zu setzen und ein passendes Business Modell und die damit verbundenen Bezahlmethoden zu wählen.

Verbreitung der App über den Windows Store oder Windows Phone-Store: Interessant hierbei ist die Bereitstellung einer App für beide Storearten. Sofern dies der Fall ist, ist zu beachten, dass die Kosten einer App einmalig gehalten werden. Somit kann eine erworbene App auf allen Windows Geräten genutzt werden, ohne für jeden Gerätetyp die App oder Funktionen die durch In-App Käufe erworben wurden, neu zu kaufen. Dies führt zu höheren Nutzerzahlen der App.

App-Informationen hinzufügen: Jede App hat seine eigene Produktseite im Windows Store. Auf dieser Seite hat der User Einsicht über alle Informationen die diese App betreffen. In der folgenden Tabelle sind die ein paar wichtige Informationen aufgelistet. Weitere Informationen und genaue Beschreibungen der Punkt sind auf der Entwicklerseite von Windows (msdn.microsoft.com) zu finden.

Info	Details
App-Name	App-Namen werden über das Verzeichnis Package/Properties/DisplayName
App-Beschreibung	Erstellen einer Beschreibung der App.
App-Feature Liste	Eine Liste aller Features die durch die App bereitgestellt werden.
Screenshots	Screenshots der App, welche dem User im Store angezeigt werden.
App-Kategorie	Wahl einer Kategorie und Subkategorie, vor dem Übermitteln der App.
Schlüsselwörter	Über Schlüsselwörter wird die App bei Suchanfragen gefunden im Windows Store.
App-Erlaubnisse	Diese werden im Manifest definiert und gibt dem User an welche Geräteerlaubnisse diese App benötigt um zu funktionieren.

Tabelle 1 – App-Informationen für den Store.

Auswahl der Märkte und Ländereinstellungen: Es ist möglich über den Windows Store über 200 Länder zu erreichen, hierzu bietet Windows die Möglichkeit die eigenen App in gewünschten Ländern bereitzustellen. Hierbei besteht die Möglichkeit, dass eine App in bestimmten Ländern nicht erlaubt wird oder es einer speziellen Altersbeschränkung bedarf, da dies durch entsprechende Gesetze in den Ländern festgelegt wurde. Des Weiteren ist es möglich durch das definieren von Sprachdateien in der App, diese für die definierten Sprachen anzubieten.

Altersbeschränkung: Um eine App in den Store zu übermitteln ist es nötig eine Altersbewertung durchzuführen. Ohne diese Altersbewertung wird die App keine Zertifizie-

rung erlangen und kann so nicht im Store angeboten werde. Einige Länder setzen sogar eine Altersbewertung durch spezielle „Rating-Boards“ voraus.

2.6.2 Zertifizierungskit für Windows-Apps

Ein App-Entwicklungsprozess besteht zu Beginn aus den gleichen Schritten. Beginnend beim Erstellen der App und des daraus folgenden Kompiliervorgangs sollte, bevor die App vertrieben wird, mittels dem von Microsoft bereit gestellten Windows-App-Certification-Kits die App validiert werden. Damit lässt sich nicht nur feststellen, ob die App für den Windows-Store geeignet ist und somit in den Store übertragen werden darf, sondern, dient auch als Hilfe um die Korrektheit der App zu bestätigen oder Fehler anzuzeigen, die dem Compiler nicht auffallen und somit verbessert werden können.

Ein paar wenige Testfälle des Zertifizierungskit (Microsoft 2014b):

- Test für abstürzende oder hängende App, um Datenverluste zu verhindern und Usability zu gewährleisten
- Test der Erfüllung technischer Anforderungen für das App-Manifest, gewährleistet ein korrekt formatiertes App-Manifest
- Test der Windows-Sicherheitsfeatures, prüft die Sicherheit der App durch Analyse des Binärcodes sowie prüfen des Systemstatus vor und nach der Installation der App
- Leistungstest, gewährleistet eine schnelle Benutzerinteraktion und agieren mit Systembefehlen, um so eine schnelle Benutzeroberfläche zu schaffen.

Zulässig sind die Test für die Betriebssysteme Windows 8, sowie 8.1. Die Möglichen Testergebnisse liefern entweder ein Bestanden oder Fehlergeschlagen.

2.6.2 Deployment in Unternehmen

Windows 8 bietet eine interessante Lösung für Unternehmen, die ihre Enterprise Software nur Nutzern innerhalb der Firma zur Verfügung stellen möchten. Dabei wäre die Verbreitung der Software über den Windows-Store keine optimale Lösung, da dieser

frei zugänglich ist und die Software somit fremden Parteien in die Hände fallen könnte. Um hierfür eine Lösung zu bieten, stellt Windows das sogenannte „Sideloading“ zur Verfügung. Mittels Sideloading wird eine App nicht vom Store heruntergeladen, sondern direkt auf den Geräten installiert auf denen die App ausgeführt werden soll.

2.6.3 Windows Store

Fällt die Entscheidung auf den Vertrieb einer Windows 8 App über den Windows Store, so sind im Vorfeld einige Details zu klären. Zumal ist eine Entwicklerlizenz ausreichend um Apps zu programmieren und zu testen. Um die App allerdings im Anschluss im Windows-Store bereitstellen zu können, ist ein zusätzliches Entwicklerkonto nötig. Microsoft unterscheidet hierbei zwischen zwei Arten von Konten: dem Einzelkonto und dem Unternehmenskonto. Dabei spielt die Art des Kontos für die Übermittlung einer App in den Store oder in den Phone-Store keine Rolle. Es unterschiedet sich das Unternehmenskonto vom Einzelkonto insofern, dass damit unternehmensspezifische Funktionen in Apps bereitgestellt werden dürfen, wie zum Beispiel der Zugriff über eine App in ein Unternehmensintranet oder die Auflistung von Desktop-Apps im Windows-Store, um so die Marketingoptionen für das Unternehmen zu steigern.

Neben den Prüfungen, die die App durch das Windows Zertifizierungskit durchläuft, werden vor dem Einreichen in den Windows-Store auch die Einhaltung der Designguide-Richtlinie sowie die Inhalte strikt geprüft. So dürfen zum Beispiel keine Inhalte, die einer Jugendfreigabe widersprechen, über den Store verbreitet werden. Einzig und allein in der Kategorie „Spiele“ hat sich diese Regelung etwas gelockert.

2.6.4 Marketing

Bevor eine Marketingstrategie gesetzt wird, wird der Entwickler dazu aufgefordert Informationen über die App bereit zu stellen. So muss jede App einen Namen, Beschreibung und Verkaufsdetails besitzen.

Bei der Wahl der Marketingstrategie ist Vorsicht geboten. Es kann durch eine falsche Wahl schnell dazu kommen, dass die App in der Masse untergeht obwohl sie, mit der richtigen Strategie anklang am Nutzer gefunden hätte.

Dabei gilt es schon während der Programmierung darauf zu achten seine App möglichst für alle Plattformarchitekturen zu erstellen.

Auch wird deutlich davon abgeraten zwei Versionen einer App, eine kostenlose und eine kostenpflichtige, zu entwickeln. Vielmehr wird von Microsoft empfohlen die In-App-Käufe in Betracht zu ziehen, um so Funktionalität freizuschalten und den User in der bereits installierten App zu behalten.

Des Weiteren sollte die App der breiten Masse zur Verfügung gestellt werden. So sollte darauf geachtet werden keine Restriktionen bei der Länderwahl oder bestimmten Märkten zu bestimmen, um so möglichst viele Interessenten zu finden.

Auch sollte man nach einer gewissen Zeit seine Strategie ändern, um so den Nutzern den kauf, falls dieser zu teuer gewesen ist, möglich zu machen oder sogar für eine bestimmte Zeitraum die App kostenlos anbieten, um so neue Kunden zu gewinnen. Auch kann man durch Sonderangebote in In-App-käufen für einen bestimmten Zeitraum, das Interesse an einer App steigern.

Es ist immer darauf zu achten die Suchergebnisse durch Tags zu fördern. Dabei sollten diese globale und auch spezifische Stichwörter enthalten.

3 Grundlagen für einen Kriterienkatalog

Dieser Abschnitt dient als Grundlagenbereich zur Entwicklung eines Kriterienkatalogs und der darauf basierenden Checkliste. Das angestrebte Ziel ist ein holistischer Entwicklungsprozess für Windows 8-Store und Phone-Apps. Zu Beginn werden zwei Softwareentwicklungsprozesse beschrieben und gegenübergestellt. Nach Abschluss der Einigung für einen Entwicklungsprozess und ihrer kritischen Betrachtung wird ein Analyse Framework, welches die Aufgabe hat das Potenzial für verschiedene Gerätearten zu bestimmen, vorgestellt. Im Anschluss wird auf das Design und die Interaktion mit den Geräten mittels User-Centered Design eingegangen. Auf Basis dieser Grundlagen wird ein Kriterienkatalog entworfen, der die Entwicklung von Windows-Store-Apps unterstützend begleitet und mit Hilfe einer Checkliste und deren Bewertungsmethode eine Einschätzung liefert, ob die zu entwickelnden Apps den Windows 8 Anforderungen genügen und den Anforderungen der Design Guidelines standhalten.

3.1 Wasserfallmodell nach Royce (1970)

Das Wasserfallmodell ist ein Vorgehensmodell im Software Engineering und wurde von Winston W. Royce im Jahr 1970 vorgestellt (JH06). Es dient zur Darstellung der Tätigkeiten bei einer Softwareentwicklung und beschreibt die komplette Lebensdauer eines Programms. Von der Analyse bis zum Betrieb und der darauffolgenden Wartung stellt jede Aktivität (siehe Abbildung 5) eine Tätigkeit im Wasserfallmodell dar. Der Durchlauf des Modells und der Aktivitäten erfolgt im Modell von oben nach unten. Mit jeder erledigten Tätigkeit erzielt man ein Teilergebnis mit dem die nächste Tätigkeit weiter arbeitet, bis hin zum fertigen Softwareprodukt. Hierbei besteht die Möglichkeit Fehler verursacht haben zu können, die erst im darauffolgenden Schritt erkannt werden. Somit ist es wichtig, dass man in frühere Tätigkeiten zurückkehren kann um diese zu beheben. Dieser Schritt wird in Abbildung 5 durch einen Zyklus modelliert, der das Wasserfallmodell abschließt. Zyklen müssen möglich gemacht werden, da in der Entwicklung Fehler so gut wie unvermeidbar sind. Somit wäre das Wasserfallmodell als striktes Einbahnstraßen-

modell nicht realisierbar. Daraus schloss man, dass Zyklen zwar geduldet aber nicht gerne gesehen werden (JH06).

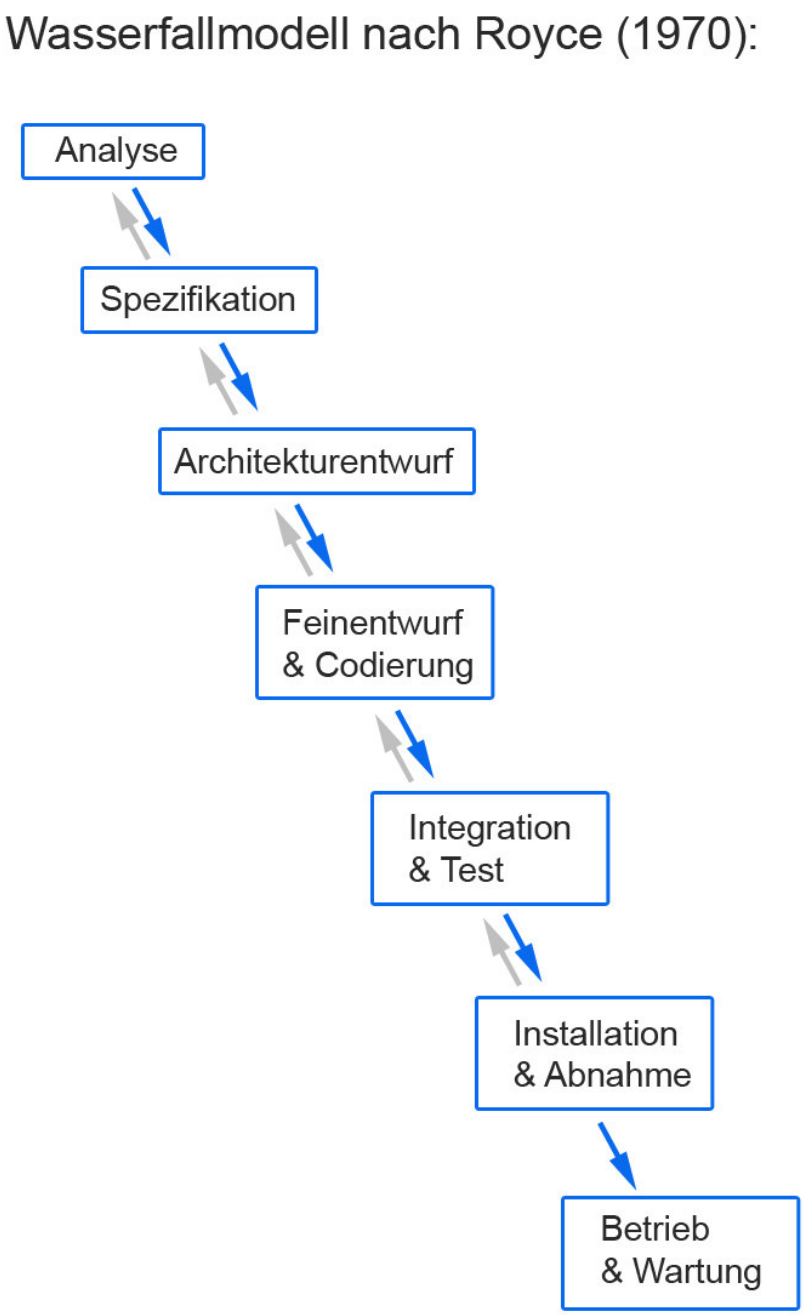

Abbildung 5 – Wasserfallmodell nach Royce (JH06).

Die folgende Tabelle zeigt die Aktivitäten eines jeden Schritts im Wasserfallmodell und beschreibt ihren Inhalt und das Ergebnis:

Aktivität	*Definition*	*Ergebnis*
Analyse	Untersuchen und verstehen des Problems. Beurteilung der Aufgaben die von der Software gelöst werden sollen.	Genaues Verständnis der Problematik und Anforderungen.
Softwarespezifikation	Anforderungen werden geordnet, dokumentiert, geprüft, ergänzt und korrigiert.	Formalisierte Beschreibung der Software.
Architekturentwurf	Festlegen einer Gesamtstruktur. Aufteilen des Projekts in Module und festlegen der Interfaces.	„Bauplan“ der Software.
Feinentwurf & Codierung	Module werden in ihrer jeweiligen Programmiersprach codiert.	Fertige Module für das Gesamtsystem.
Integration & Test	Die fertigen Module werden zusammengesetzt (integriert) und getestet.	Fertige Software.
Installation & Abnahme	Software wird installiert und vom Kunden Abgenommen. Spezieller Test aller Funktionen durch den Kunden.	Software bereit zur Inbetriebnahme.
Betrieb & Wartung	Software wird am Einsatzort installiert und genutzt. Im späteren verlaufen auffallende Fehler oder neue Anforderungen werden durch Wartungsprojekte behoben oder erstellt.	Fertiges Produkt.

Tabelle 2 – Wasserfallmodell: Die verschiedenen Aktivitäten, deren Definition und Ergebnis (JH06).

3.2 Agile Softwareentwicklung

Bei agiler Softwareentwicklung ist das Ziel die Softwareentwicklung möglichst flexibel zu gestalten. Dabei berücksichtigt dieses Vorgehensmodell sowohl technische als auch soziale Probleme der Softwareentwicklung. Agile Softwareentwicklung macht sich hierbei ein iteratives Vorgehen zu Nutze und umgeht so einen hohen bürokratischen Aufwand und kommt somit auch mit weniger Regeln aus.

Agile Softwareentwicklung besteht aus vier wesentlichen Bestandteilen:

Agile Werte: Bilden das Fundament. Diese Grunderkenntnisse wurden im Frühjahr 2001 als agiles Manifest formuliert (JH06):

1. Menschen und Interaktionen sind wichtiger als Prozesse und Werkzeuge.
2. Funktionierende Software ist wichtiger als umfassende Dokumentationen.
3. Zusammenarbeit mit dem Kunden ist wichtiger als die ursprünglich formulierten Leistungsbeschreibungen.
4. Eingehen auf Veränderungen ist wichtiger als das Festhalten an einem Plan.

Agile Prinzip: Bildet den Leitsatz der Agile Softwareentwicklung. Diese Prinzipien beschreiben sowohl den Umgang mit dem Kunden, als auch Veränderungen während der frühen bis hin zur späten Entwicklungsphase. Des Weiteren beinhaltet das Agile Prinzip die Lieferung von funktionierender Software und deren technisch einwandfreien Zustand in möglichst geringen Abständen. Dabei enthalten diese Prinzipien auch den Umgang im Team und die Zusammenarbeit mit Fachexperten.

Agile Methoden: Beschreiben die Methoden, die während der Softwareentwicklung genutzt werden, um den Aufwand möglichst gering zu halten. Ein Beispiel für solche Methoden ist die Paarprogrammierung bei der während der Programmierung jeweils zwei Programmierer an einem Rechner arbeiten.

Agile Prozesse: Sind eine Sammlung von Prozessen, die auf den drei vorhergehenden Punkten basieren. Agile Prozesse versuchen den Bürokratischen Aufwand zu minimieren und stellen soziale Aspekte in den Vordergrund. Ihr Ziel ist es so früh wie möglich ausführbare Software zu erstellen und diese immer in kurzen Abständen mit dem Kunden

zu besprechen, um so flexibel auf dessen Wünsche einzugehen. Diese Kundennähe steigert auch die Zufriedenheit des Kunden und lässt Änderungen bis in die späte Entwicklungsphase zu.

3.2.1 Agile Prozesse

Zwei der beliebtesten und am häufigsten verwendeten Agile Prozesse sind Extreme Programming (XP) und Scrum. Diese unterscheiden sich im Wesentlichen sehr stark. Während XP sich vorrangig auf die Umsetzung und das Lösen einer Programmieraufgabe fokussiert, stellt Scrum das Management und den Prozess von Softwareentwicklung in den Vordergrund.

3.3 Agile versus Wasserfall

Das traditionelle Wasserfallmodell steht auf dem Grundbaustein einer genauen Planung bevor mit der Entwicklung begonnen wird und die strikte Einhaltung dieser, während der kompletten Entwicklungsphase. Dies setzt voraus, dass alle Wünsche und Anforderungen bekannt sind und sich während der linearen Prozesse des Wasserfallmodells und deren sequentiellen Ausführung nicht mehr ändern. Dieses Modell setzt einen einwandfreien Einsatz der erforderten Technologie voraus. Allerdings führt dies zu wesentlichen Nachteilen: Probleme werden erst spät erkannt. Dies führt zu verschobenen Release Terminen oder zum Weglassen von Funktionen die bis dato nicht fertig gestellt werden konnten oder nicht die gewünschte Funktion aufweisen. Selbst bei einfachen Updates oder Änderungen wird eine aufwändige Planung vorausgesetzt, was zu steigenden Entwicklungskosten führt. Vom Kunden gewünschte Veränderungen während einer Entwicklungsphase sind nur schwer umzusetzen, da diese auch wieder eine neue Planung voraussetzen. Anders als beim Wasserfallmodell bauen Agile Softwareentwicklungsprozesse hier auf eine iterative und empirische Herangehensweise. Dadurch beobachtet der Agile Prozess laufend die aktuelle Situation der Entwicklung und kann so Informationen transparent und effizient darstellen und ändern. Dies setzt im Gegensatz zum Wasserfallmodell keine Vorhersagen voraus, sondern eine zyklische Beobachtung unbekannter Probleme, welche durch die Frequenz der Zyklen analysiert und bearbeitet

werden kann. Durch dieses Vorgehen werden die oben genannten erzeugten Probleme des Wasserfallmodells so gut wie vollständig behoben.

Durch diese getroffene Erkenntnis wird im weiteren Verlauf und der Erstellung des Kriterienkatalogs die Agile Softwareentwicklung als Grundbaustein genommen. Dies hat den Vorteil, dass Änderungen durch Probleme oder fehlenden Funktionen iterativ beseitigt werden können und man effizient durch den Kriterienkatalog Windows 8 Apps entwickeln kann.

3.4 User-Centered-Design

In einer technisch geprägten Umgebung steht meistens die Funktionalität einer Software im Mittelpunkt (JH06). Um den Endnutzern die Nutzung des Systems leichter zu machen, sollte sie in den Fokus gesetzt werden. Um dies zu realisieren geht man nach dem Prinzip des User-Centered-Design vor, welches die Aufgabe besitzt interaktive Software so zu gestalten, dass sie über ein hohes Maß an Usability und User Experience verfügt. Dabei wird zwischen diesen zwei Begriffen stark differenziert (BN09), wobei die genaue ISO 9241-210 (ISO) vorgibt, dass Usability vor allem die Effektivität und Effizienz eines System beschreibt und die User Experience sich vorrangig mit den Eindrücken vor, während und nach dem Interagieren mit dem System beschäftigt.

Das User-Centered-Design Prozess basiert auf einem iterativen Vorgehen welches aus vier Phasen besteht. Diese bauen auf Prinzipien auf wie (GJC85):

- *Dem frühen Fokus auf den User*: Somit kann der Designer besser auf die Fähigkeiten des Nutzers eingehen und die Software dementsprechend gestalten.
- *Es müssen empirische Werte gesammelt werden:* Hiermit können bereits im frühen Entwicklungsstadium, mittels Prototypen oder Simulationen, Verhaltensmuster wie Leistung und Reaktion, beobachtet, aufgezeichnet und analysiert werden.
- *Die Voraussetzung, dass der Designprozess iterativ ist:* So können gefundene Fehler bei Usertests, durch Re-Design behoben werden. Dieser Schritt wird so oft wiederholt bis das Ergebnis zufriedenstellend ist.

Hieraus entwickelt sich ein vier-Phasen-Konzept:

In der ersten Phase steht die *Analyse des Nutzungskontexts,* welcher im Grunde die Beschaffung von Informationen über die Benutzer beinhaltet. Es werden die Benutzer in Nutzerprofile zusammengefasst und zusammen mit den Zielen, Arbeitsabläufen und die Umgebung, in der sie sich befinden, analysiert. Im nächsten Schritt werden durch die gewonnen Ergebnisse, aus der Analyse der Benutzer, die *Anforderungen definiert.* Diese dienen als Richtlinie für die weitere Umsetzung des Projekts. Darauf folgt die *Umsetzung eines Prototypen und Entwurfs.* Dieser kann aus Designdokumenten, Mockups oder Papier-Prototypen bestehen. Der Prototyp dient im Anschluss des Entwurfs der *Evaluation,* in der mit Hilfe der Nutzer, die Prototypen wiederholt besprochen und ausprobiert werden. Nach diesem Schritt ergibt sich ob man die Anforderungen der Nutzer erfüllt hat oder man das Konzept redesignen sollte.

3.5 Analyse Framework

Die Beliebtheit von mobilen Geräten steigt in Unternehmen immer mehr an. So werden, um die Produktivität zu steigern, Firmenangestellte häufiger mit mobilen Geräten, wie Tablets oder Smartphones ausgestattet. Nun stellt sich der Firma die Frage welche Technologie für eine bestimmte Aufgabe genutzt werden soll. Dabei kann die Firma zwischen PCs und Laptops als stationäre IT Systeme oder Smartphone und Tablets als mobile Touchscreen basierende Geräte wählen.

Dieser Abschnitt und das Analyse Framework basiert auf der Arbeit von Eva Hoos, Christoph Gröger, Stefan Kramer und Prof. Dr.-Ing. habil. Mitschang.

Das Analyse Framework dient zur Entscheidungsfindung und beantwortet die Frage welches Gerät eine bestimmte Aktivität am besten verrichten kann. Dabei beachtet es wesentliche Voraussetzung:

- Potential der mobilen Technologie.
- Art des mobilen Geräts.
- Ganzheitlicher Blick auf den Prozess.

Das Analyse Framework dient damit als Grundlage zur Analyse und den ersten Schritten eines konzeptionellen Entwurfs. Weiter bietet es Hilfestellung in der Frage für welche Geräte die Windows 8 App entwickelt werden soll.

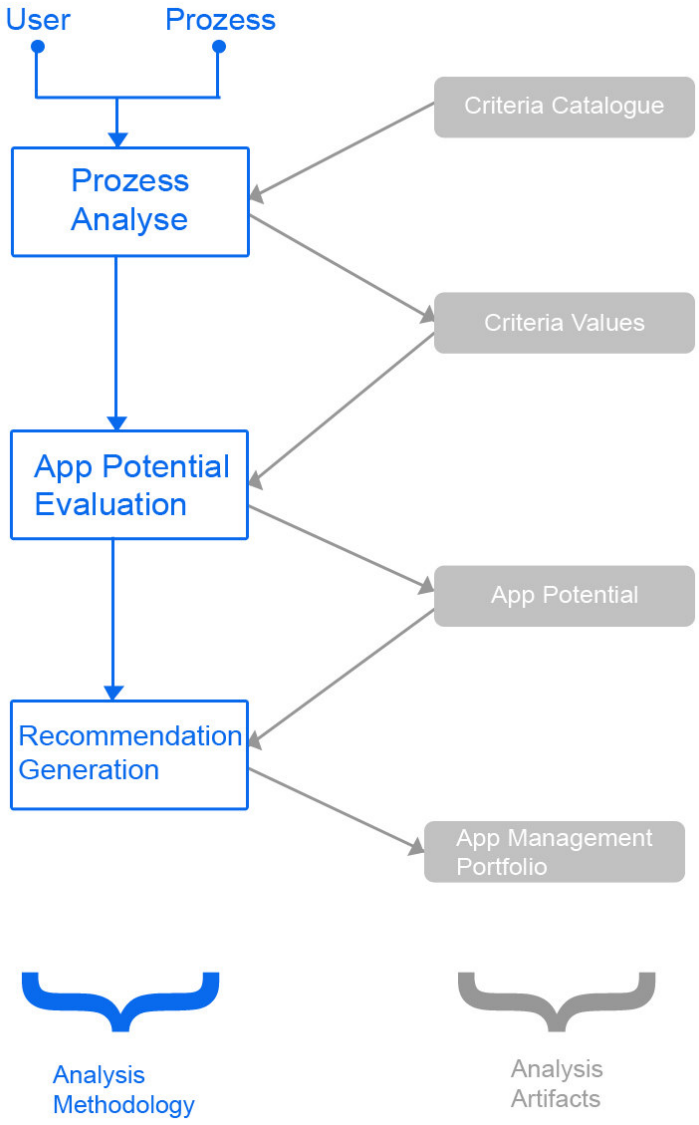

Abbildung 6 – Ablauf des Analyse Frameworks (HGKM).

3.5.1 Anwendung

Das Analyse Framework besteht im Wesentlichen, wie in Abbildung 6 zu sehen ist, aus zwei Grundbausteinen: der Analyse Methodik und verschiedener Analyse Werkzeuge.

Zu Beginn strebt der User oder ein bestimmter Prozess eine Prozessanalyse mit Hilfe eines speziell dafür entworfenen Kriterienkatalogs an. Das Ergebnis dieser Analyse sind die daraus folgenden Criteria Values, welche ein Mapping aus den erworbenen Ergebnissen beschreiben. Dieses Ergebnis besteht aus einer Prozessanalyse durch den Criteria Catalogue. Dabei wir ein zwei dimensionales Wertepaar x_{AppCap}, X_{MobPot} welches das App Potential bestimmen und mit damit der Prozess evaluiert wird. Dies geschieht mit folgender Formel in Kombination mit einer „Scoring Matrix“ aus welcher die Kriterien und ihre Ergebnisse einen Wert zugewiesen bekommen:

$$AppPotential = \left(x_{AppCap}, X_{MobPot}\right)$$

mit:

$$x_j = \sum_{c \in D_j} s(C = k_c) * w_c$$

$$where\ j \in \{AppCap, MobPot\}$$

Score	*3*	*2*	*1*
Task	High	Medium	Low
Actor	High	Medium	Low
Frequency	Often	Regularly	Rarely
…	…		

Tabelle 3 – Auszug aus der Scoring Matrix. (HGKM)

Der Ablauf dieses Mappings beginnt mit der Entscheidung ob ein Kriterium dem Mobilitätspotential, welches beschreibt ob die Aktivität eine Auswirkung auf die Mobilität eines Gerätes hat, oder der Computing Power, welche die Leistungsfähigkeit einer Aktivität beschreibt, zugewiesen wird. Beispielsweise ist die Erweiterung einer Applikation durch Sensoren ein Kriterium, das der Mobilität zugeschrieben wird. Somit wird es der Dimension D_{MobPot} zugewiesen. Durch die Zuweisung dieses Kriteriums der *Dimension: Mobilization Potential,* kann im Anschluss durch die Formel der entsprechende Wert dieser Dimension hinzugefügt werden. So kann, wie in Tabelle 2 zu sehen ist, eine Frequency im Criteria Catalogue mit „Regularly" gewertet werden. Somit ist die Score $s(Frequency = Regularly) = 2.$ Fernen kann der Actor mit Low belegt werden, daraus folgt dass die Summe der Dimension in der Actor abgelegt ist um $s(Actor = Low) = 1$ erhöht wird. Sofern ein bestimmtes Kriterium besonders wichtig für eine Aufgabe ist, kann diese durch w_c speziell gewichtet und somit erhöht werden.

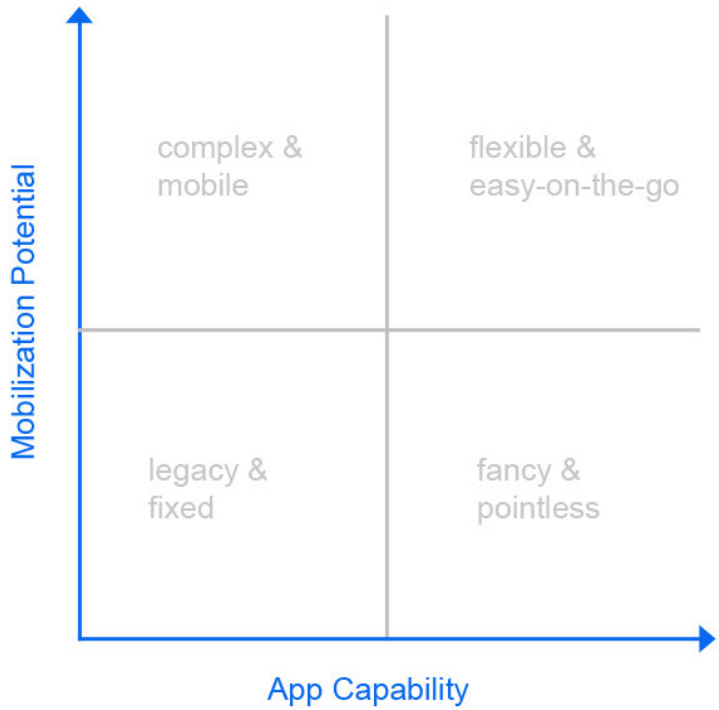

Abbildung 7 – App Management Portfolio, (HGKM).

Im nächsten Schritt „Recommendation Generation“ werden die erzielten Ergebnisse des App Potentials in das App Management Portfolio positioniert, wie in Abbildung 7 zu sehen ist. Dort werden die Ergebnisse grafisch dargestellt und in vier Bereiche eingeteilt, diese sind aus Abbildung 7 zu entnehmen. Diese geben an welcher Kategorie die App, durch die Kriterien und das daraus folgende App-Potential erreicht hat. Somit kann bestimmt werden, welche Aktivitäten eine App bereitstellen sollte und diese im Entwicklungsprozess höhere Aufmerksamkeit zugewiesen werden kann.

4 Mobile und Desktop App-Entwicklung

Im Wesentlichen unterscheiden sich mobile Apps und solche für Desktop und Tablets bereits in ihrem Betriebssystem. Dies hat zur Folge, dass man, für eine App die für beide Gerätetypen konzipiert werden soll, einen doppelten Programmieraufwand hat. Mit dem neusten Update auf Windows 8.1 basiert, anders als seine Vorgänger, Windows Phone nicht mehr auf Windows CE-Kernel sondern benutzt den gleichen Kernel wie Windows 8 und Windows RT. Mobile Apps werden mit dem Update auf Windows 8.1 für Microsoft Windows Phone Betriebssysteme konzipiert, welches der Nachfolger des Microsoft Windows Mobile System ist. Dabei basieren Windows Phones auf dem Windows NT-Kernel und zählen somit nicht mehr zur Windows Mobile Linie. Durch das Gleichstellen der Kernel bietet Microsoft dem Entwickler die Möglichkeit Apps mit nur wenigen Designanpassungen und geringem Programmieraufwand sowohl für Smartphones mit Windows Phone 8.1 und Desktopgeräten zu erstellen.

4.1 Universal-Windows-App

Dieses neu eingeführte Konzept wird als „Universal-App“ bezeichnet und wird in allen von Windows bereitgestellten Programmiersprachen unterstützt. Um eine Universal-App zu erstellen benötigt es nicht mehr als die aktuellste Entwicklungsumgebung von Visual-Studio welches das entsprechende Update enthält.

Erstellen:

Beim Erstellen einer Universal-App ist es nach der Wahl der Programmiersprache wichtig, eine Universal-App Projektmappe zu erstellen. Wie man in Abbildung 8 erkennen kann, wurde beispielhaft Visual C# als Programmiersprache gewählt. Im sich darunter öffnenden Auswahlbereich erkennt man die Kategorien: Windows Phone-Apps, Windows-Apps und Universal-App. Nach der entsprechenden Wahl kann man sich ein gewünschtes Template erstellen und nachdem der Projektname festgelegt wurde, wird sowohl das Projekt erzeugt, als auch die damit verbundenen Projektmappen.

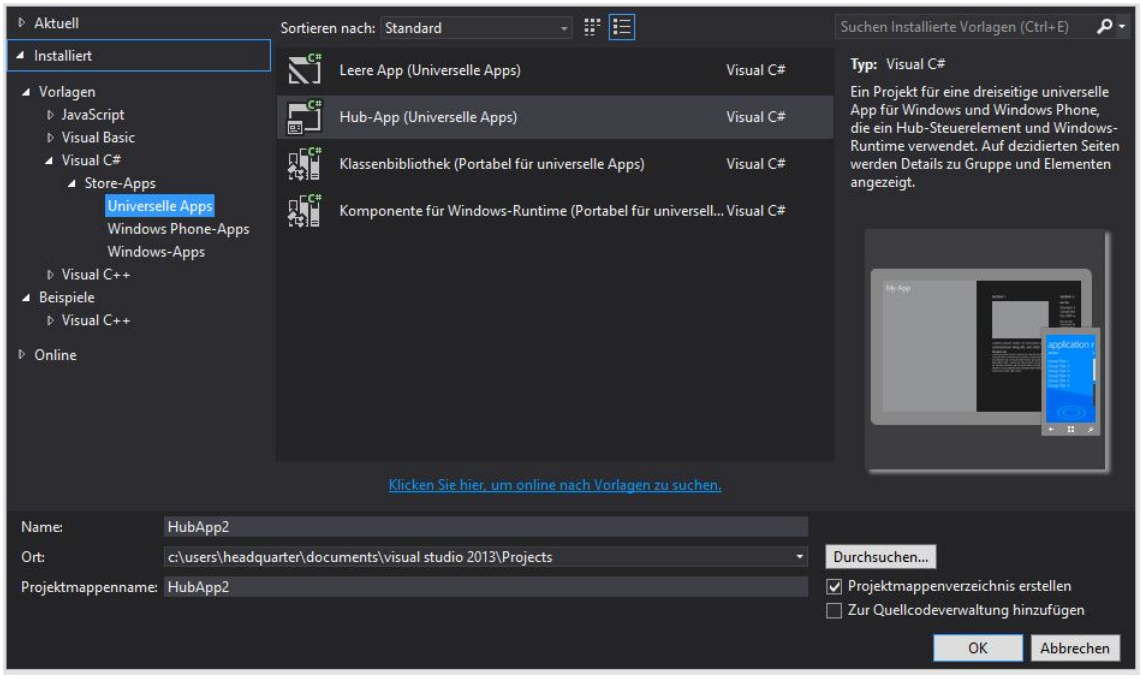

Abbildung 8 – Visual Studio Universal App erstellen.

Struktur:

Sofern das Projekt erstellt wurde strukturiert sich die Projektmappe, wie in Abbildung 9 zu sehen ist, in drei Kategorien.

- ProjektName.Windows(Windows 8.1) – enthält die Projektdaten für Apps die auf größere Geräte abzielen, wie zum Beispiel einen Desktop PC, Laptop oder Tablet.
- ProjektName.WindowsPhone(Windows Phone 8.1) – enthält die die Daten für die Erstellung von Smartphone Geräten.
- ProjektName.Shared – enthält Projektdaten welche von beiden Projekten benutzt werden können. Sowohl Phone als auch Desktop App haben auf diesen Bereich Zugriff.

Durch die Trennung der Projektmappen in die beiden Plattformen ist es möglich für Windows und Windows-Phone die entsprechenden Design- und Seitenstrukturen festzulegen. Genauso ist es möglich Klassen, die nur von einer Plattform benutzt werden dürfen, nur für die entsprechenden Geräte zu definieren, da sie beispielsweise Sensoren verwendet wie GPS oder Lagesensoren, welche bei Desktopanwendungen nicht vor-

handen sind, in den entsprechenden Ordnern zu erstellen und so sicher zu stellen dass nur diese Plattform genannten Klassen nutzen darf.

Möchte man Daten wie Bilder, Klassen, Datenmodelle für beide Typen bereitstellen, so werden sie in der Shared-Mappe erstellt und verwaltet. Beide Plattformen haben vollen Zugriff auf diesen Bereich und teilen sich die entsprechenden Daten. Dies ist vor allem von Vorteil um dem Entwickler Zeit und Ressourcen zu ersparen, da er sonst einen Doppelaufwand für beide Plattformen hätte.

Verteile:

Sofern das Projekt vollständig erstellt wurde und bereit zum Veröffentlichen ist, wird aus der einzelnen Projektmappe zwei Pakete geschnürt, zum einen für den Windows Store und zum anderen für den Windows Phone Store. Diese Pakete können an die jeweiligen Stores übermittelt werden.

Dieses neue Konzept der App Entwicklung bietet sowohl für den Entwickler als auch für den Endnutzer wesentliche Vorteile.

Dem Entwickler ist somit ein Tool in die Hände gereicht worden, mit dem er gleichzeitig für mehrere Plattformen Ressourcensparend programmieren kann, was ihm Zeitersparnis bringt und den Aufwand verschiedene Projekte zu organisieren, so gering wie möglich hält. Auch kann die Verwaltung von Käufen und Benachrichtigungen plattformübergreifend bearbeitet werden.

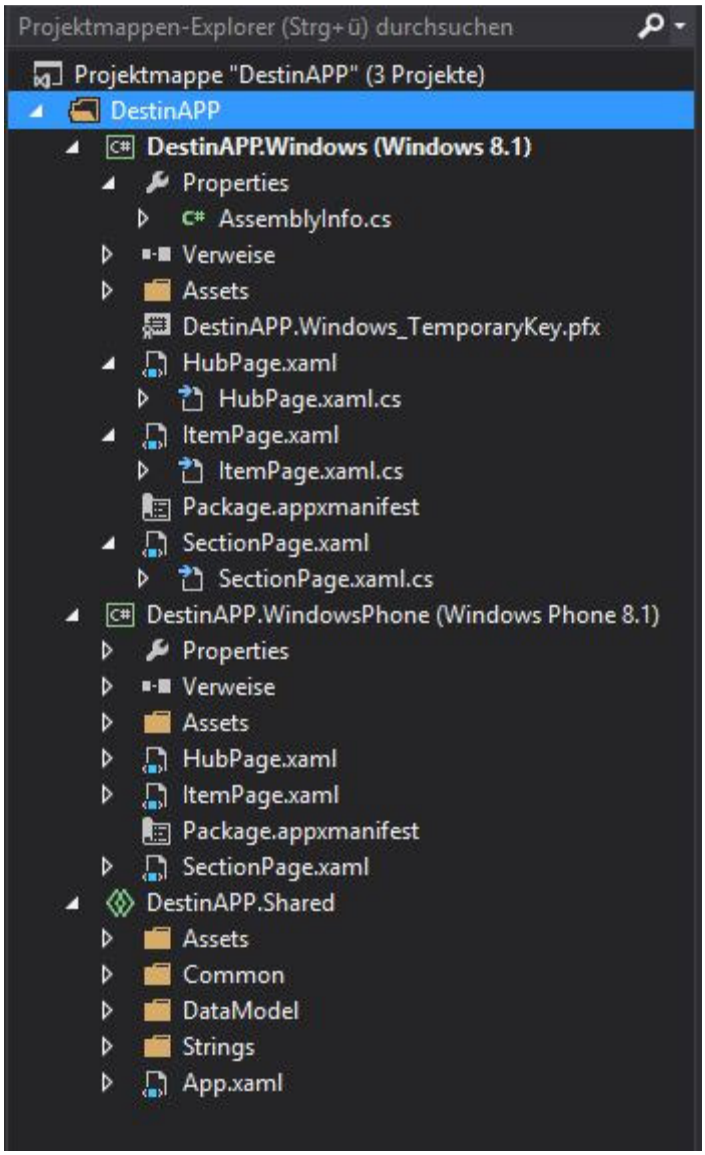

Abbildung 9 – Universal App Ordnerstruktur

Genauso kann der Kunde durch eine gemeinsame App-Identität in den verschiedenen Stores mit nur einem Konto, eine App installieren und nutzen, ohne diese für beide Plattformen kaufen zu müssen. Dies gilt auch für In-App-Käufe, welche für alle Geräte sofort verfügbar wären und so nicht auf einer anderen Plattform erneut gekauft werden müssen.

5 Kriterienkatalog zur Windows 8 App-Entwicklung

Im Folgenden wird ein Kriterienkatalog zur Unterstützung des Entwicklungsprozess von Windows 8 Store Apps beschrieben. Die aufgeführten Kriterien betrachten sowohl technische Aspekte als auch Designkonzepte hinsichtlich einer guten Mensch-Computer Interaktion, basierend auf dem Konzept der Windows 8 Designpatterns. Dabei zielt der Kriterienkatalog auf die Einhaltung der App-Zertifizierungskriterien und der Design Guidelines von Windows 8 ab. Das Ziel ist es, einen Kriterienkatalog zu entwickeln, welcher alle wesentlichen Aspekte für eine gute Entwicklung abdeckt und alle wichtigen Anforderungen betrachtet.

5.1 Kriterienkatalog

Der Kriterienkatalog verhält sich in der Nutzung wie eine Art Checkliste, da beide für die hier beschriebenen Zwecke den gleichen Aufbau besitzen mit dem der Entwickler sukzessive durch den Entwicklungsprozess einer Windows Store App geführt wird. Dabei enthält Katalog, Rahmenbedingungen, als auch, Designpatterns und einen Schrittweise ablaufenden Deploymentprozess, sowohl für die öffentliche Nutzung als auch zur Nutzung für Unternehmen im internen Gebrauch. Auch wird im dritten Teil der Checkliste die Möglichkeit einer Universal App beschrieben und Schrittweise dargestellt.

Durch das erfolgreiche absolvieren des Kriterienkatalogs als Checkliste, ist es möglich, effizient eine App zu erstellen, die den Anforderungen von Windows 8 genügt und im Store veröffentlich oder für den internen Gebrauch von Unternehmen genutzt werden kann.

I. Rahmenbedingungen:

In diesem Bereich werden die Rahmenbedingungen zur Entwicklung von Windows Store-App betrachtet. Diese beinhalten sowohl Entwickler- als auch Endnutzeraspekte und bilden den Grundbaustein Entwicklung und Nutzung von Windows Store-Apps.

Der strukturelle Ablauf der Rahmenbedingungen ist als Flowchart in Abbildung 10 – Flowchart für die Rahmenbedingungen zu sehen. Die Reihenfolge spielt hierbei eine wichtige Rolle, sofern diese Kriterien nicht oder nur gering eingehalten werden, kann es zu schweren Komplikationen in der Entwicklung und Nutzung der Apps kommen oder die Entwicklung sogar ganz verhindert werden:

I.a Vorbereitung

Gemeinsame Geräteanforderungen:

Gerätekompatibilität: Sind die Geräte auf denen die App ausgeführt werden soll und solche auf denen Entwickelt wird, Windows 8 kompatibel.

Installation/Update Windows 8.1: Wurde bereits Windows 8.1 installiert oder das Update von Windows 8 auf 8.1 durchgeführt.

[Optional]Mitarbeiterschulungen: Müssen Entwickler und Nutzer zusätzlich geschult werden.

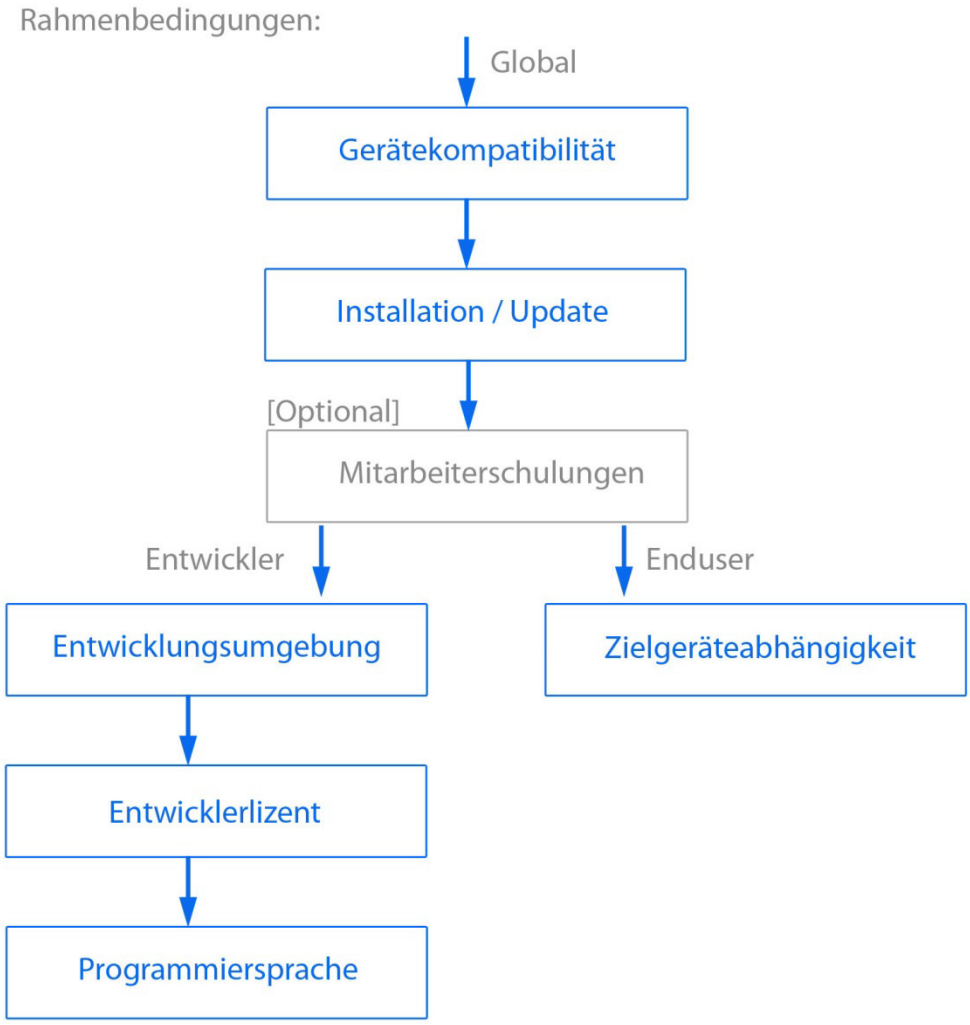

Abbildung 10 – Flowchart für die Rahmenbedingungen

Entwickler:

Für weitere Informationen, siehe hierzu Grundlagenteil Abschnitt 2.3 Entwicklungsumgebung und 2.4 Programmiersprachen.

Entwicklungsumgebung: Ist die neuste Entwicklungsumgebung von Visual Studio auf dem Entwicklergerät aufgesetzt.

Entwicklerlizenz: Ist ein Entwickleraccount für Windows 8 angelegt worden. Siehe hierzu Grundlagenteil Abschnitt 2.3 Entwicklungsumgebung.

Wahl der Programmiersprache: Entscheidung ob mit JavaScript/HTML, VisualBasic/C#/C++ mit XAML oder C++mit DirectX entwickelt werden soll.

Endusergeräte:

Zielgeräte: Entscheidung, ob für Desktop, Phone oder/und Tablet entwickelt werden soll. Eine Entscheidungsfindung wäre mit dem Analyse Framework möglich, siehe hierzu Abschnitt 3.5 Analyse Framework.

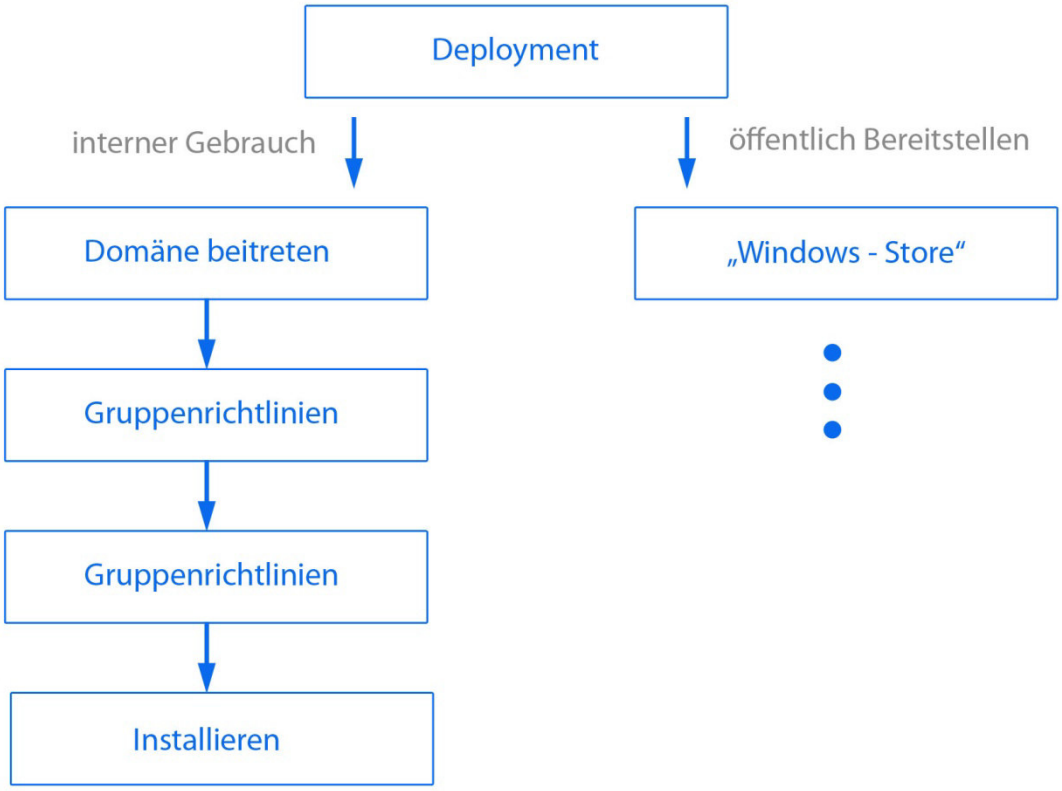

Abbildung 11 – Flowchart Deployment

I.b Deployment

Im Vorfeld muss festgelegt werden ob die App für den öffentlich Gebrauch oder Firmen internen Nutzern bereitgestellt werden soll. Für eine genauere Beschreibung des Vorgangs, siehe im Grundlagenteil 2.6 Deployment.

Fall 1, Interner Gebrauch:

Computerdomäne: Liegen die zu Benutzenden Geräte in derselben Domäne wie die Windows Enterprise Version.

Gruppenrichtlinieneinstellungen: Aktivieren der „Installation aller vertrauenswürdigen Anwendungen zulassen“ - Funktion.

Zertifikate: App mit Zertifikat signiert und mit vertrauenswürdigem Stammzertifikat verknüpft.

Installieren: Entweder über ein Windows-Image oder einzeln auf die Nutzer PC verteilen. Für eine genauere Beschreibung siehe im Grundlagenteil 2.6 Deployment.

Fall 2, Öffentlicher Gebrauch:

Für den öffentlichen Gebrauch sind einige, nicht unerhebliche, Schritte nötig. Hierfür sollte der Entwickler sich bereits im Vorfeld Gedanken über einen passenden Namen und Beschreibung der Store-App machen, sowie die jeweilige Verkauf Strategie. Diese Punkte sind Entwicklungsrelevant. Desweitern gibt es Punkte, wie das Erstellen von Screenshot, die erst am Ende der Entwicklung Sinn ergeben.

Für den Schrittweise ablaufenden Prozess, siehe hierzu den weiterführenden Punkt „Onlinestellen der App“.

Programmiertechnisch zu beachten sind folgende Punkte:

In-App-Käufe: Entschließt man sich für In-App-Käufe, sollte die entsprechende Logik im Quelltext implementiert werden.

Werbung: Sofern die App Werbung enthalten muss. An welcher Position sollte diese Angezeigt werden und welchen Werbepartner nimmt man hierzu.

Lightversion/Vollversion: Sofern es eine kostenlose Lightversion der App geben soll. Darf diese keine voll implementierten Funktionen haben, sondern werden aus dem Quelltext entnommen. (So kann selbst nach einem erfolgreichen Cracken der kostenlosen Lightversion diese dennoch nicht voll genutzt werden).

II. Design

Mit Windows Apps kam auch wie im Kapitel „Einführung in Windows 8“ bereits beschrieben, ein neues Designkonzept. Dieses beruht auf verschiedenen Designprinzipien, welche bei Einhaltung, während der Entwicklung, sowohl ein möglichst effizientes als auch an die Windows 8 Umgebung angepasstes Ergebnis liefern. Dabei wird der Fokus stark um den User gelegt und die damit eingehende Interaktion in den Mittelpunkt gestellt.

Diese Designkonzepte sind bei der kompletten Gestaltung und in den Design Guidelines von Windows 8 (Microsoft - User experience guidelines) zu finden. Diese sind zu beachten und dienen als Leitfaden.

Um die künstlerische Freiheit nicht einzuschränken werden in diesem Teil des Kriterienkatalogs lediglich die Patterns vorgestellt und die damit resultierenden Ergebnisse beschrieben. Die Wahl der zu nutzenden Patterns liegen im Ermessen des Designers und können zwischen verschiedenen Anwendungsgebieten variieren.

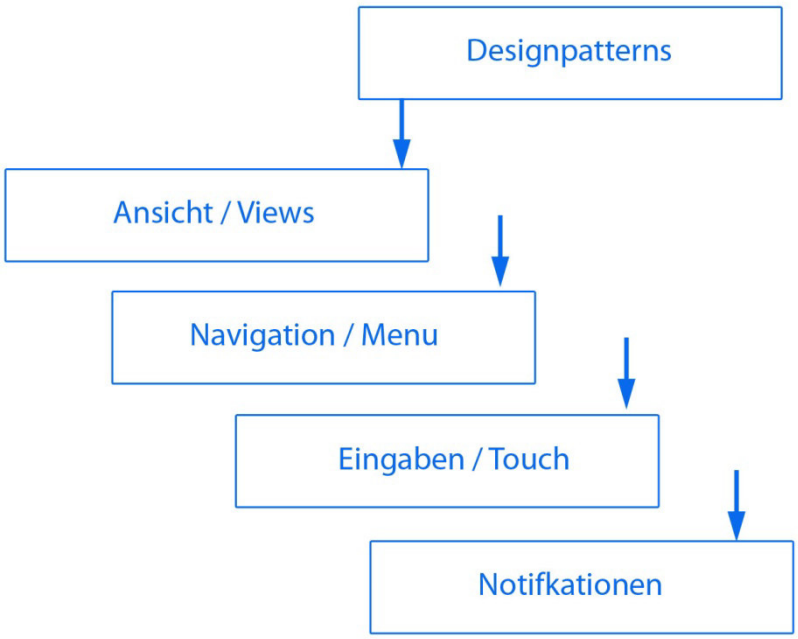

Abbildung 12 – Flowchart der Designpatterns.

II.a Ansicht

Dieser Abschnitt behandelt die Frage der Ausrichtung des Bildschirms, dieser kann sowohl fest für eine Ausrichtung sein oder sich automatisch an beide anpassen. Des Weiteren gibt es drei verschiedene Bildschirmbereiche die in Windows 8 genutzt werden können.

Landscape/Portrait:

Bei der Wahl der Ansicht spielen verschiedene Faktoren eine wichtige Rolle. Zum einen wählt der Designer im Vorfeld welche Bildschirmausrichtung genutzt werden soll, zum anderen muss er beachten welche Bildschirmbereiche er ansprechen möchte und mit welchem Layout Pattern er dies optimal realisieren kann. Bei der Wahl einer Bildschirmausrichtung (Bildschirmorientierung) unterscheidet man zwischen Landscape und Portrait.

Die Wahl kann entweder auf eine von beiden Optionen fallen oder es wird die Programmierung so optimiert dass sich die App bei einer Veränderung der Geräteausrichtung automatisch anpasst. Dies gelingt insofern nur, wenn hierfür die passenden Restriktionen in XAML- oder CSS- File getroffen wurden. Vorteile einer automatischen Ausrichtung sind das bessere und flexiblere Darstellen der Inhalte. Im Gegenzug macht es bei speziellen Apps nicht immer Sinn beide Ausrichtungsmöglichkeiten zu nutzen und man beschränkt die App auf eine Ansicht.

Snapped-, Fill-, Full-View:

Des Weiteren bietet Windows 8 die Möglichkeit den Bildschirmbereich in der Landscape-Ansicht, wie in Abbildung 13 - Verschiedene Ansichtsmöglichkeiten (in Anlehnung an den Windows 8 Designguide) (Microsoft - User experience guidelines).zu sehen ist, in drei Bereiche zu unterteilen. Diese sind in durch verschieden Seitenverhältnisse (siehe Tabelle 4 – Darstellungsverhältnis Snapped-, Fill- und Full-View.) definiert. Dies hat zum Vorteil, dass man durch das Aufteilen des Bildschirms, zwei Apps gleichzeitig darstellen kann und die Nutzung sich insofern erleichtert, dass es nicht mehr nötig ist die App ständig zu wechseln. Genauso ist eine Kommunikation dieser Apps über entsprechende Programmierschnittstellen möglich.

Full-View	*Fill-View*	*Snapped-View*
1:1	2:3	1:3

Tabelle 4 – Darstellungsverhältnis Snapped-, Fill- und Full-View.

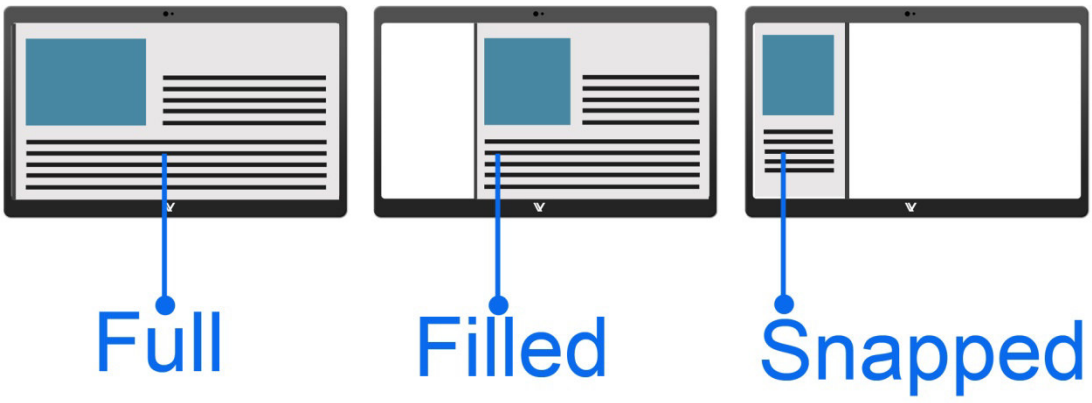

Abbildung 13 - Verschiedene Ansichtsmöglichkeiten (in Anlehnung an den Windows 8 Designguide) (Microsoft - User experience guidelines).

Layout und Struktur

Nach der Entscheidung welche Bildschirmorientierung gewählt und für welche Bildschirmbereiche die Store-App entwickelt werden soll, folgt die Wahl eines passenden Layout Patterns und der Entwicklung mittels einer Grid-Struktur.

Fluid/Static Layout:

Für den Designer ist es wichtig möglichst effizient die Bildschirmgröße zu nutzen und dem User ein handliches Design zu bieten. Frei von unnötigem Zoomen oder Scrollen. Um die Usability zu gewährleisten, bietet Windows 8 die Möglichkeit durch Benutzung von prozentualen Größenangaben die App automatisch an verschiedene Bildschirmgrößen anzupassen. Diese Art des Layouts wird als Fluidlayout bezeichnet und steht im Kontrast zum Staticlayout, bei dem die Größenangaben fest zugewiesen werden.

Für Spezialfälle bei Apps würde ein statisches Layout durchaus Sinn machen, allerdings ist dies für die breite Masse meistens nicht der Fall. Die Wahl eines Layout Patterns liegt

hier wieder im Ermessen des Designers und hängt von der Aufgabe der entsprechenden App ab.

Grid-Struktur:

Eine Grid-Struktur, definiert ein Rastersystem, um Elemente besser im System anzuordnen und bei verschiedenen Bildschirmgrößen durch das Fluidlayout neu strukturieren und skalieren zu können. So wird ein bestimmter Bereich in Blöcke und die Blöcke in Reihen und Spalten unterteilten, welchen die entsprechenden Inhalte zugewiesen werden. So wird beim Öffnen der App in auf einem kleineren oder größeren Bildschirm, das Rastermuster so restrukturiert, dass alle Elemente inhaltlich in ihrer erwünschten Ordnung bleiben.

II.b Navigation

Die Inhaltsstrukturierung einer App basiert auf einem hierarchischen System. Dadurch wird die Informationsdarstellung über verschiedene Abstraktionslevel realisiert (Siehe hierzu Abbildung 14 – Beispielhafte Darstellung der App-Bar aus dem App Verzeichnis von Windows 8.). Zusätzliche Optionen oder Einstellungen können über versteckte Menus an den Rändern des Screens, durch den Mauszeiger oder bei Touch Geräten durch eine Swipe-Bewegung, erreicht werden.

Splashscreen: Für jede App sollte ein Splashscreen definiert werden, welcher das Logo enthält. So wird beim Starten der App der Splashscreen angezeigt und die App wird im Hintergrund geladen.

Hub- Page: Definieren der Startseite der App. Diese beinhaltet eine Einführung in die App, so wie eine passende Beschreibung und eine Teaser-Übersicht der verschiedenen Sektionen. Dabei ist die Hub-Page in unterschiedliche Hub-Sektionen unterteilt, welche auf dem Prinzip der Grid-Struktur basieren.

Section-Page: Für jede Kategorie sollte eine Section-Page als Template geladen werden, welche die einzelnen Inhalte und Elemente repräsentiert hinzukommen Zusammenfassungen der entsprechenden Themen in der jeweiligen Kategorie und Teaser für die einzelnen Elemente.

Detail-Page: Diese Art der Page dient als Template für den eigentlichen Inhalt zu einem bestimmten Thema. Dabei werden hier die entsprechenden Inhalte zu einem Thema dargestellt und präsentiert.

[Optional]App-Bars: Diese ermöglichen dem User einen schnellen Zugriff auf Befehle welche nicht direkt sichtbar in der App erscheinen dürfen, da sie sonst beispielsweise den Inhalt einschränken oder sogar verdecken würden.

[Optional] Charms: Das Einfügen von Charm-Menus wird genutzt um weitere Redundanzen im Informationsfluss zu vermeiden. Erreichbar sind diese über den rechten oder optional linken Rand des Bildschirms.

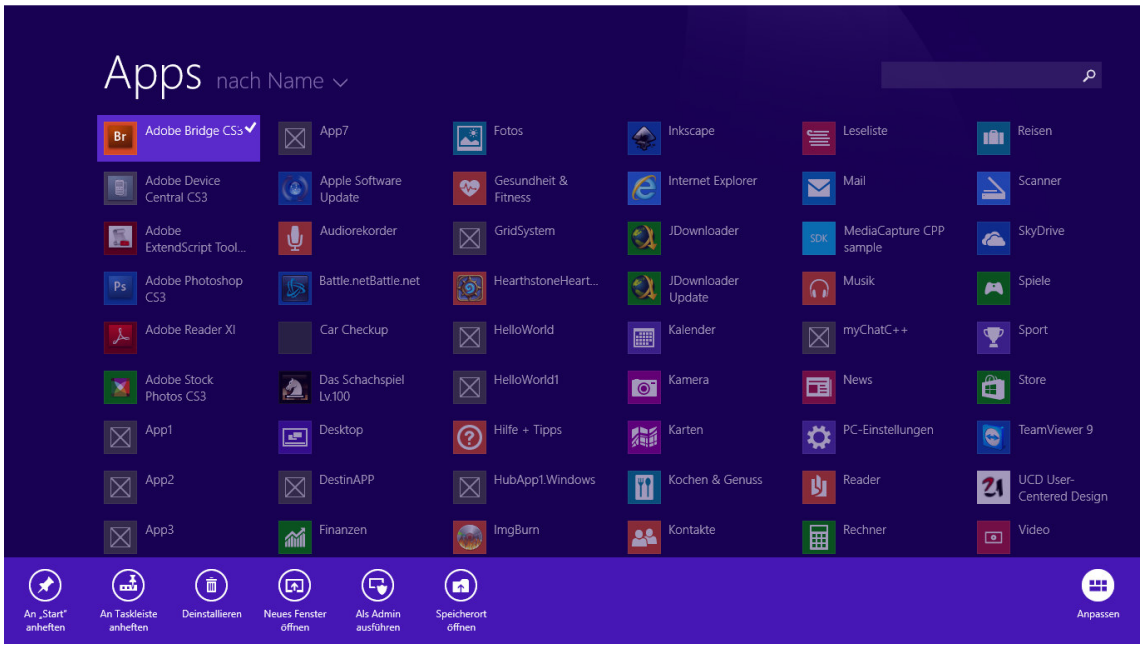

Abbildung 14 – Beispielhafte Darstellung der App-Bar aus dem App Verzeichnis von Windows 8.

II.c Touch

Nutzung der Windows 8 Touch-Language für Tablets und entsprechenden Bildschirmen. Mit Einhaltung dieser Patterns werden Interaktionen des Users mit dem Gerät an die Windows 8 Standard Bedienung angepasst. Dies ist besonders wichtig, da dem User ein vertrautes Gefühl mitgeteilt wird und er sich so nicht aufs Neue in die App einlernen muss. Außer denen hier vorgeführten Punkten gibt es noch einige weitere, welche in den Windows Design Guidelines zu finden sind und ebenfalls beachtet werden sollten.

Touch Input: Die Größe der Elemente einer App muss für Touch Interkationen entsprechend größer gewählt und definiert werden. Hierfür wird eine minimale Größe von 7mm x 7mm (Microsoft - User experience guidelines) in den Windows Guidelines empfohlen.

Touch Feedback: Bei einer Touch Interaktion sollte das Gerät ein Feedback geben. Diese können haptische Feedbacks, wie ein kurzes Vibrieren, sein, so wie ton- oder visuelle- Feedbacks.

Inhalte durchblättern durch Touch: Anstatt sich durch eine tief gehende Seitenstruktur durch zu klicken, sollte eine flüssige Navigation durch eine breite Hub- Section- oder Item-Page ermöglicht werden. Hierbei muss der User nicht ständig eine neue Seite laden, sondern erreicht die Inhalte durch das einfache scrollen nach rechts. So bekommt er ohne erheblichen Klickaufwand alle Information präsentiert und kann bei gewünschten mehr Information die tiefe des Informationslevels wechseln.

Menus durch Touch: Der User erreicht Menus mit einer einfachen Touch Bewegung vom äußeren rechten oder linken Rand und dem unteren oder oberen Rand. So gewährleistet das Windows Design eine schnelle Menu Führung sowie minimalen Auswand des User um dieses zu erreichen.

[Hinweis] Weiter Optionen und Interaktionsarten sind in den Guidelines zu finden (Microsoft - User experience guidelines).

II.d Bereitstellen von Bildern

Die folgende Tabelle 5 – Bilderauflösung für Windows 8 Apps enthält eine Übersicht aller benötigten Bilderarten, die in einem App-Paket einfügen werden sollten. Das App-Paket sollte Bilder für den Startbildschirm, so wie ein Store-Logo enthalten. Des Weiteren sollte ein Begrüßungsbildschirm (Splashscreen), welcher beim Starten einer App angezeigt wird, enthalten sein. Tabelle 5 – Bilderauflösung für Windows 8 Apps enthält die zu jeder Bild Art die dazugehörigen Größenangaben, Datentypen und ob die Bilder für eine Zertifizierung im Store notwendig sind. Um keine Verzerrungen der Anzeigebilder zu erzeugen sollte für jede skalierte Version, ein dementsprechend passendes Bild erstellt werden. Größenunterschiede können nicht durch skalieren des Originalbilds erzeugt werden, allerdings ist die Bereitstellung für die verschiedenen Bildformate in den meisten Fällen optional.

Name	**Größen (Pixel)**	**Datentyp**	**Notwendig für Zertifizierung**
Store-Logo	100 %: 50 x 50 140%: 70 x 70 180 %: 90 x 90	.png, .jpg, .jpeg	100%: 50 x 50
Square150x150-Logo	80 %: 120 x 120 100 %: 150 x 150 140%: 210 x 210 180 %: 270 x 270	.png, .jpg, .jpeg	nein
Square30x30-Logo	80 %: 24 x 24 100 %: 30 x 30 140%: 42 x 42 180%: 54 x 54	.png, .jpg, .jpeg	100 %: 30 x 30
Wide310x150-Logo	80 %: 248 x 120 100 %: 310 x 150 140 % 434 x 210 180 % 558 x 270	.png, .jpg, .jpeg	Nein, aber empfohlen.
Square310x310-Logo	80 %: 248 x 248 100 %: 310 x 310 140 %: 434 x 434 180 %: 558 x 558	.png, .jpg, .jpeg	Nein.
Square70x70-Logo	80 %: 56 x 56 100 %: 70 x 70 140 %: 98 x 98	.png, .jpg, .jpeg	Nein.

	180 %: 126 x 126		
Logo für die Infoanzeige	100%: 24 x 24 140 %: 33 x 33 180 %: 43 x 43	.png, .jpg, .jpeg	Nein.
Begrüßungsbildschirm	100%: 620 x 300 140%: 868 x 420 180%: 1116 x 540	.png, .jpg, .jpeg	100%: 620 x 300.

Tabelle 5 – Bilderauflösung für Windows 8 Apps

II.e [Optional] Animationen und Notifikationen

Animationen und Benachrichtigung sind je nach Gebrauch optional, da sie abhängig von den gewünschten Anforderungen der App sind. Fehlerbehandlungen sollten dennoch nach Windows 8 App Konventionen behandelt werden, da sie bereits ein fester Bestandteil der Infrastruktur sind und ein eigenmächtiges Implementieren von Pop-Ups oder derartigen Animationen, zu einer fehlerhaften Darstellung der App führen oder ungewünschte Verschiebungen im Grid-System hervorrufen können.

Context menus: Werden dazu genutzt um Aktionen die der User bei Text oder UI-Elementen durchführen kann zu definieren. Beispielsweise: Copy, Paste, Cut. Dieses Menu ist auf bis zu fünf Elemente beschränkt und ist mit dem herkömmlichen Klicken der rechten Maustaste zu vergleichen.

Message Dialoge: Dies sind direkte Pop-Up Felder, die eine direkte User Interaktion erfordern um diesen vor weiteren Eingaben abzuhalten. Dies ist nützlich falls Fehler auftreten oder die App eine explizite Bestätigung benötigt. Dabei wird der Hintergrund verdunkelt und das Dialogfenster in den Vordergrund gesetzt.

Flyouts: Ähnlich wie das Charm-Menu werden Flyouts, genutzt um Benutzereingaben zu erfassen, Details einzelner Elemente anzuzeigen oder User zu einer Bestätigung aufzufordern. Gestartet werden diese durch ein Tippen oder Klicken und wieder geschlossen sobald ein Bereich außerhalb des Flyouts angeklickt wird.

Toasts: Sind App-übergreifende Benachrichtigungen, als Pop-Up im oberen rechten Eck des Bildschirms, informieren diese den User über Ereignisse einer anderen App.

Beispielsweise erhält man Toasts beim Erhalten neuer E-Mails, Eingehende VOIP Anrufe oder Textnachrichten sowie Kalendertermine.

III. Universal App

Dieser Bereich beinhaltet das Thema Universal App und kann wie bereits die Kriterien Kategorie „Rahmenbedingungen“ als Checklisten Struktur angesehen werden. Dabei gliedert sich der Ablauf mehr auf die Erstellung und Struktur einer Universal App als auf das eigentliche Programmieren ab, da dies wie bei einer herkömmlichen App behandelt wird.

Für eine genaue Beschreibung und mehr Informationen zu Universal Apps dient Abschnitt 4 im Grundlagenteil.

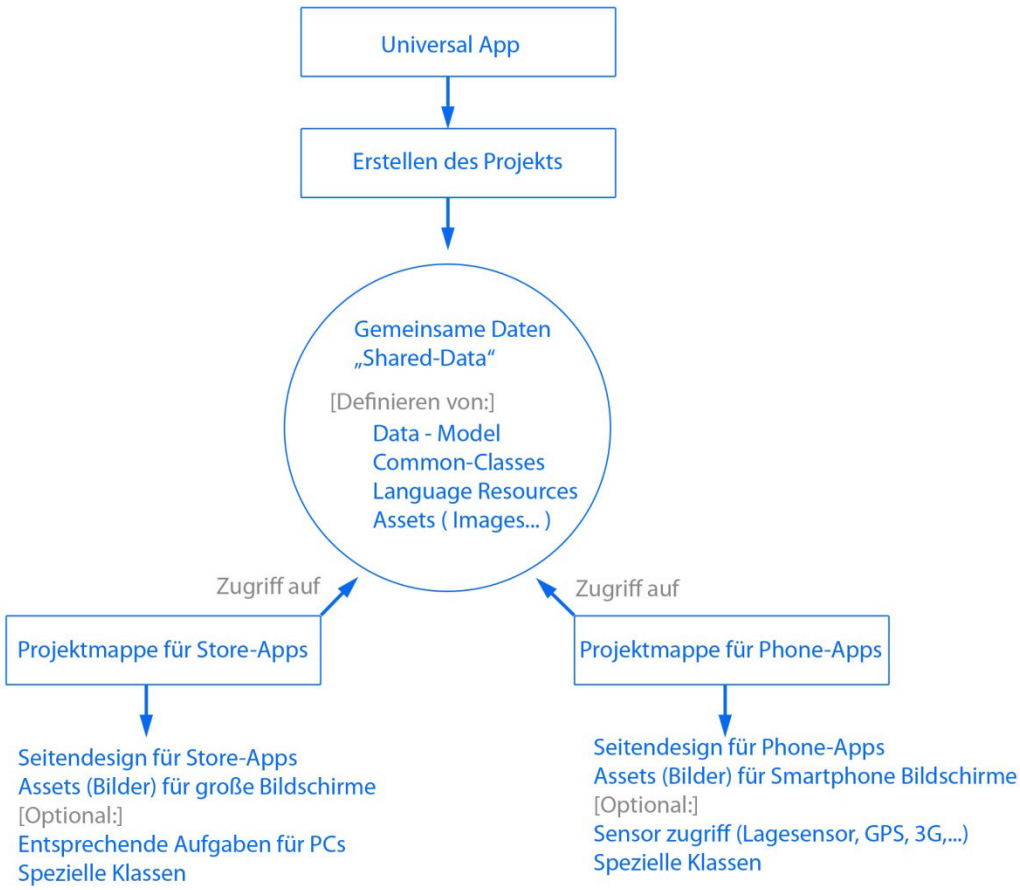

Abbildung 15 – Universal App Projektmappen Zugriffe.

Erstellen einer Universal App: Nach dem Starten von Visual Studio, muss ein neues Projekt angelegt werden und im Unterpunkt „Store-Apps“, der gewählten Programmiersprache, die Kategorie Universelle App ausgewählt werden. Anschließend muss ein Projektname vergeben werden und mit „Ok“ das Projekt erstellt.

Definieren gemeinsamer Languagefiles: Anlegen von Sprachressourcen in der Shared-Projektmappe. Diese werden sowohl von der Windows Store-App als auch der Phone-App genutzt und dienen zum einen zur besseren Verwaltung von Titel und Textinhalten und der Erstellung von mehrsprachigen Apps.

Definieren gemeinsamer Daten sowie Logikstrukturen und Helperklassen: Gemeinsame Daten Modelle wie JSON, XML oder Datenbanken werden in der Shared-Mappe gespeichert und verwaltet. Klassen werden auch aus dieser Mappe aufgerufen und in die Phone-App oder Store-App eingebunden.

Definieren gemeinsamer Grafischer Objekte: Gemeinsame Assets wie Bilder und Videos werden ebenfalls in der Shared-Mappe abgelegt.

Erstellen des Designs für Windows 8 Apps: Erstellen der Pages speziell für die Windows 8 App und definieren des Designs in der Projektmappe „Windows“. Diese Mappe enthält spezielle Daten auf die nur die Store-App zugreifen kann. Hinzu kommen Assets die für entsprechend große Bildschirmgrößen bereitgestellt werden sollen.

Erstellen des Designs für Windows 8 Phones: Design und Pages werden in der „WindowsPhone“ Mappe erstellt. Alle in dieser Mappe definierten Dateien können ausschließlich von WindowsPhone genutzt werden.

[Optional] Klassen oder Datenstrukturen für einzelne Plattformen erstellen: Insofern eine Plattform aufgabenabhängig andere Klassen oder Datenstruktur enthalten muss, so sind diese in der entsprechenden Projektmappe zu erstellen und zu verwalten.

[Optional] Ansprechen von Sensoren für Phone, Tablet, PC/Notebook: Wahl und Nutzung von Sensoren, wie zum Beispiel die Kamera, der Lagesensor, GPS oder mobiles Internet über 3G oder LTE, für die jeweiligen Geräte.

V. Onlinestellen der App

Dieser Abschnitt des Kriterienkatalogs beschreibt das Onlinestellen der App. Diesbezüglich werden sowohl Entwicklerkonto und die nötigen Informationen, die im App-Store bereitgestellt werden müssen betrachtet. Des Weiteren wird die Wahl eines Geschäftsmodells betrachtet und das damit eingehende Marketing. Für einen genaueren Ausführung der einzelnen Schritte und eine Beschreibung des Dashboards Siehe Kapitel 2.6 Deployment.

Erstellen eines Entwicklerkontos: Dies ist nötig um Projekte in das DevCenter zu übertragen, so kann eine App Zertifizierung erlangt werden und anschließend die App im Store bereitgestellt werden. Es wird zwischen Einzelkonto oder Unternehmenskonto unterschieden.

App-Spezifische Informationen der App: Eintragen wichtiger Informationen der App wie, Name, Beschreibung und hinzufügen von Screenshots in das Dashboard.

Wahl des Geschäftsmodells: Festsetzen eines Geschäftsmodells und die zugehörigen Preise. Dabei hat der Entwickler die Option eine kostenlose Testversion anzubieten oder feste Preise, Transaktionen von Dritten so wie In-App-Käufe.

Werbung/Anzeigen: Erstellen eines Microsoft Advertising PubCenter-Kontos, Dadurch erhält man ein AdUnitId-Element welche in Quelltext verankert wird und über die die Kosten abgerechnet werden. Angezeigt wird die Werbung im AdControl-Element in der App selbst.

Upload: Es muss ein komprimiertes App Paket der Anwendung im Dev Center-Portal hochladen. Des Weiteren muss eine App-Beschreibung hinzugefügt werden und mögliche Hinweise für den Tester.

Zertifizierung: Falls Fehler vorhanden sind müssen diese behoben werden. Anschließend muss das Projekt erneut zur Zertifizierung bereitgestellt werden. Sollten keine Fehler auftreten, ist die App bereit in den Store geladen zu werden und steht ab diesem Zeitpunkt funktionsfähig im Store zu Verfügung.

6 DestinAPP

Dieses Kapitel befasst sich mit der Entwicklung der DestinAPP. DestinAPP ist die Abkürzung für „Decision Support to Identify Beneficial Apps“ und stellt eine Realisierung, des in Abschnitt 3.5 vorgestellten Analyse Framework dar. Im weiteren Verlauf wird die Analyse der Anforderungen aus Sicht des Kunden vorgestellt und der Entwurf eines Prototypen von der Struktur bis hin zur Datenverwaltung.

6.1 DestinAPP

Im Rahmen dieser Bachelorarbeit soll eine Windows 8 App mit dem Namen „DestinAPP“ implementiert werden. Diese App hat die Aufgabe, das in Abschnitt 3.5 vorgestellte Analyse Framework zu realisieren. Voraussetzungen für die Entwicklung der App sind eine klare Agile Prozessstruktur, sowie den User in den Fokus der Entwicklung zu stellen. Dabei wird der Entwicklungsprozess mittels, den im vorherigen Abschnitt vorgestellten, Kriterienkatalog begleitet und evaluiert.

DestinAPP ist die Abkürzung für „Decision Support to Identify Beneficial Apps“ und stellt eine Realisierung, des in Abschnitt 3.5 vorgestellten Analyse Framework dar. Dabei soll es möglich sein Prozesse anzulegen, welche aus verschiedenen Aktivitäten bestehen, die mit dem Analyse Framework auf ihr App-Potenzial untersucht werden und mittels einer Diagrammansicht grafisch dargestellt und begutachtet werden können.

Dies ermöglicht dem Nutzer die schnelle Analyse verschiedener Aktivitäten die in ihre Gesamtheit einen Prozess ergebe. So kann schnell entschieden werden ob es für die Entwicklung Sinn macht ein Prozess für mehrere Gerätetypen zu entwickeln oder ob dies einzelnen vorenthalten bleibt, da die Bedienung nur erschwert oder nicht möglich wäre. Auch lässt sich daraus ableiten, ob eventuell einzelne Aktivitäten als eigenständige Prozess ausgelagert werden sollen.

6.2 Entwicklungsprozess

Im Folgenden wird der Entwicklungsprozess von der Spezifizierung der Anforderungen bis hin zur Analyse und Entwurf der DestinAPP begleitet.

6.2.1 Anforderungen an die App

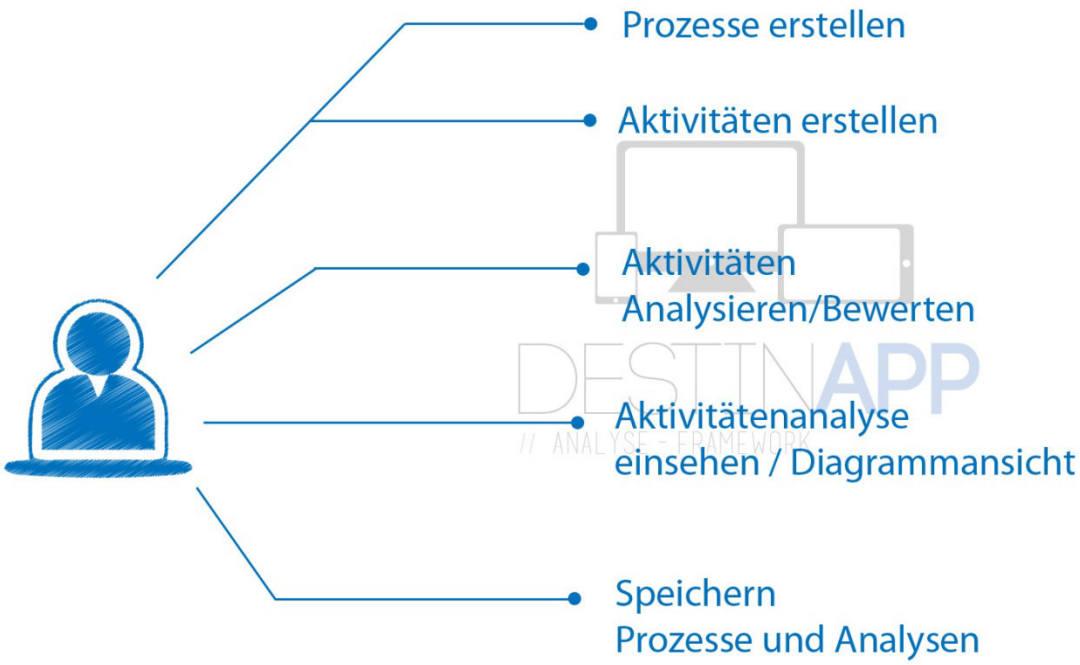

Abbildung 16 – Anforderungen an die DestinAPP.

Wie in Abbildung 16 – Anforderungen an die DestinAPP.zu sehen ist, werden an die DestinAPP einige Anforderungen gestellt. Zum einem soll es möglich sein, Prozesse zu erstellen, welche eine beliebige Anzahl an Aktivitäten enthält. Jede Aktivität soll mittels der Logik des Analyse Frameworks Analysiert und Bewertet werden. Jede erstellte Aktivität soll persistent, das bedeutet dauerhaft abgespeichert werden und nicht nur für jede Session verfügbar sein. Es sollen alle Prozesse in Listenform angezeigt werden, so können diese Strukturiert bearbeitet oder eingesehen werden. Bereits erstellte Aktivitäten und deren Bewertung sollen über die Prozesse in denen sie angelegt worden sind erreichbar sein und eingesehen werden können. Jede Aktivität kann eine Vorgänger- und Nachfolgeraktivität besitzen, dies soll eindeutig realisiert werden.

6.2.1 Analyse und Entwurf

Es soll eine für Windows 8 funktionsfähige App entwickelt werden, welche Ihren Fokus auf ein User-Centered Design legt und dabei die Windows 8 App Struktur nicht vernachlässigt. Die App soll aus einer Hub-Page, welche die Startseite bildet, bestehen. Diese soll das Projekt vorstellt und einen kurzen Überblick auf die zuletzt angelegten Prozesse geben. Ferner kann man über die Hub-Page zu einer vollständigen Liste aller Prozesse navigieren, über die man zu den entsprechenden Aktivitäten und zu deren Detailansicht gelangt, um dort das Evaluierungsdiagramm der einzelnen Aktivitäten auslesen zu können. Genauso soll es möglich sein über die Hub-Page einen neuen Prozess anzulegen oder einem bereits bestehenden Prozess eine Aktivität mit Vorgänger- und Nachfolgeraktivität zu definieren. Sofern eine neue Aktivität erzeugt wird gelangt man zu einer Detailansicht in der sich die Evaluierungsfragen befinden und ausgefüllt werden können. Beim Speichern der Evaluierungsfragen wird die Aktivität ausgewertet, dem Prozess hinzugefügt und die Diagramme in der Detailansicht der jeweiligen Aktivität erzeugt.

Die Datensicherung erfolgt über eine SQLite Datenbank, die lokal benutzt und verwaltet wird. Dabei bilden Projekt, Aktivität und Kriterien die jeweiligen Entitäten die mit den entsprechenden Daten gefüllt und abgefragt werden.

Abbildung 17 – Screenshot aus Visual Studio 2013.

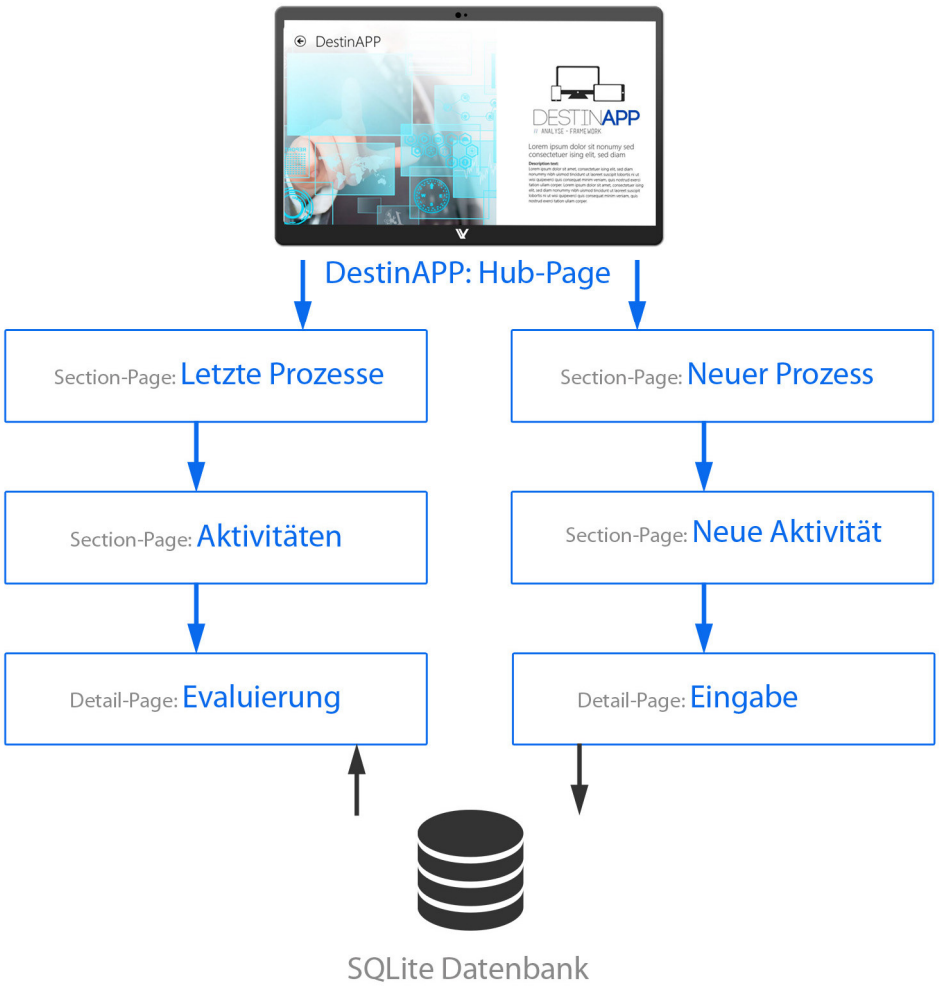

Abbildung 18 – Projektstruktur DestinAPP.

Die Projektstruktur gliedert sich, wie in 8 zu sehen ist, in zwei Bereiche.

Zum einen kann der Nutzer einen Prozess erstellen und eine beliebige Anzahl von Aktivitäten erstellen, aus denen der entsprechende Prozess besteht. Zum anderen ist es möglich die Ausgewerteten Informationen zu betrachten. Dabei bildet die Datenbank das zentrale Kernstück.

Des Weiteren wurde für das Analyse ein Logo entwickelt welches zum einen für eine gestalterische Informationsdarstellung in der Hup-Page dient und zum anderen, in angepasster Form als Splashscreen genutzt wird. Das Logo und ein kleiner Introbereich bilden den Startpunkt der App wie in Abbildung 17 – Screenshot aus Visual Studio 2013.

zu sehen ist. Beim Scrollen nach rechts, folgen weitere Informationsteaser und eine kurze Übersicht der letzten Projekte.

7 Evaluation der DestinAPP mittels Kriterienkatalog

In diesem Kapitel wird die DestinAPP fachgerecht, anhand des erstellten Kriterienkatalogs bewertet. Dabei wird sukzessive jedes Kriterium im Katalog als Checkliste verwendet und an der zu erstellenden App (DestinAPP) angewandt. Dabei gibt es hier ein genaues Mapping zu den erstellen Kriterien im Kriterienkatalog, sodass jeder Schritt im Detail nachvollzogen werden kann. Auch wird jeder Schritt und eine mögliche Entscheidung genau beschrieben. Dieses Kapitel, setzt Kapitel 5 und ein Verständnis der Grundlagen von Windows 8 voraus.

I Rahmenbedingungen:

Den Startpunkt bilden die Rahmenbedingungen, mit denen die App erstellt wird und für ein Deployment über den Windows Store oder als Branchen App innerhalb eines Unternehmens vorbereitet.

I.a Vorbereitung

Gerätekompatibilität:	Das Entwicklergeräte ist Windows 8 Kompatibel und alle Treiber und Funktionen wurden Sachgemäß installiert.
Installation/Update Windows 8.1:	Das Update wurde durchgeführt.
[Optional] Mitarbeiterschulung:	Dem Entwickler ist der Umgang mit Windows 8 vertraut.
Entwicklungsumgebung:	Es ist die aktuelle Version: Visual Studio 2013 installiert.
Entwicklerlizenz:	Eine Entwickleraccount wurde erstellt und eine Lizenz wurde erhalten.

Programmiersprache:	Die ausgewählte Programmiersprache ist C#, da diese in Kombination mit den Voraussetzung der DestinAPP aufgrund der einfachen Einbindung einer SQLite Datenbank, exakt die Anforderungen trifft. Des Weiteren hat der Entwickler bereits gute Kenntnisse in C# was das Erstellen eines passenden Layouts und die Implementierung der Logik vereinfacht.
Zielgeräte:	Das erste Konzept wird als Universal App erstellt, die Implementierung folgt aber fürs erste nur für Desktop und Tablets.

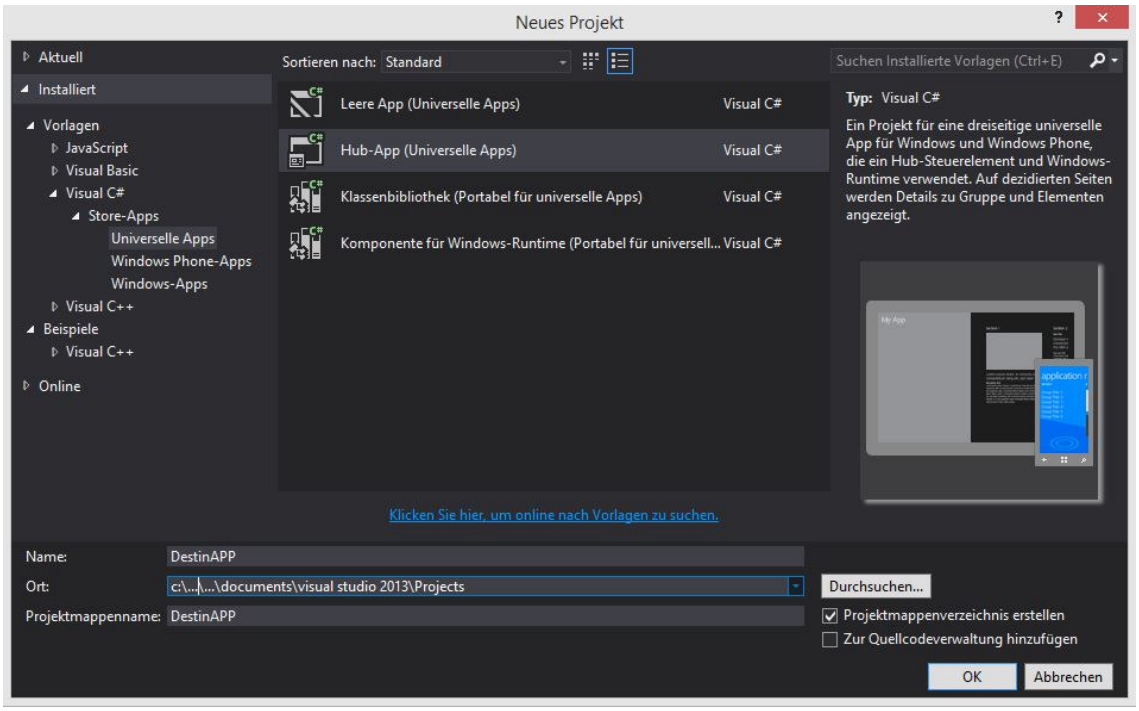

Abbildung 19 – Erstellen eines neuen Projekts.

I.b Deployment

Die App wird im späteren Verlauf als interne Branchen App genutzt und nicht über den Store vertrieben. Hierfür muss für die Weiterentwicklung entsprechende Konfiguratio-

nen vorgenommen werden. Die Kriterien können somit, an dieser Stelle, noch nicht bewertet werden.

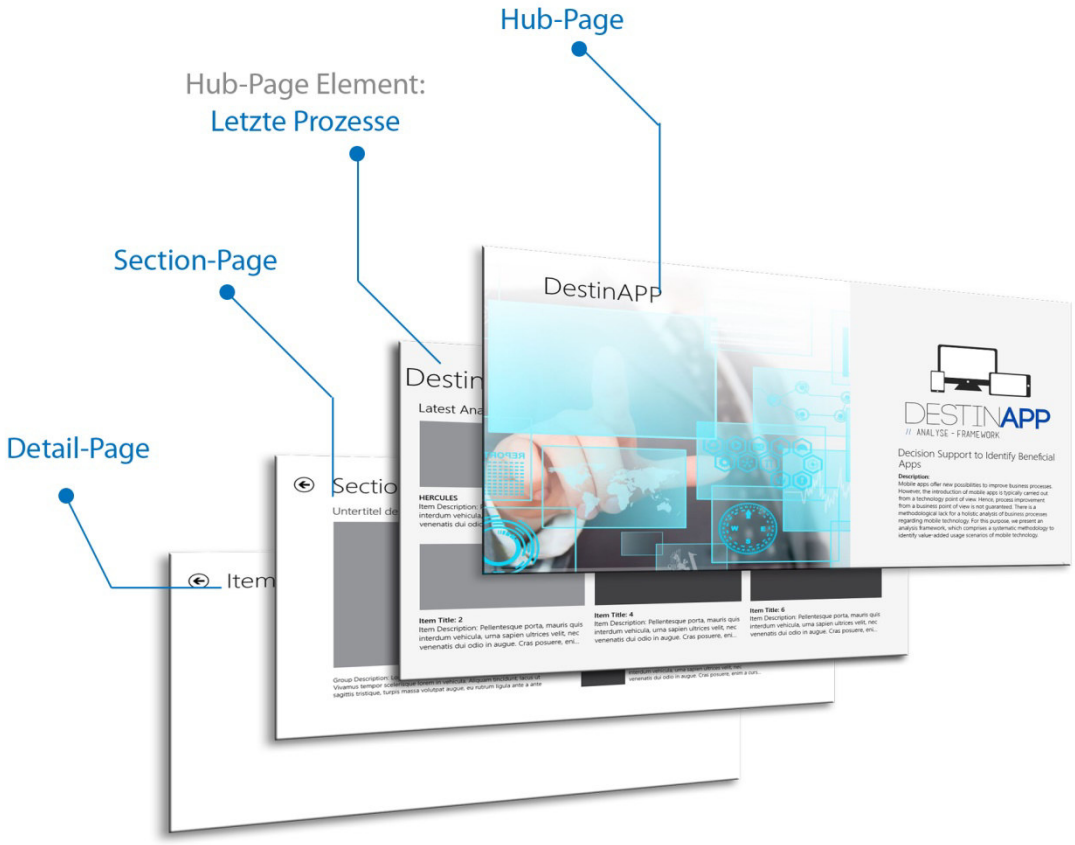

Abbildung 20 – Hierarchische Struktur der DestinAPP.

II Design

II.a Ansicht

Landscape/Portrait:	Die App wird nur in Landscape Ansicht nutzbar sein. Dieses bietet wie Abbildung 21 zu sehen ist eine breiteren Bildschirmbereich und eine harmonischere Bedienung, da alle Elemente in der horizontalen ausgerichtet sind.
Snapped-, Fill-, Full-View:	Es wird keine besonderen Anpassungen für die unterschiedlichen Views in der Konzeption geben – Dies kann aber in einer tiefergehenden Programmierung

	hinzugefügt werden, da alles fachgerecht nach einem Grid-Layout definiert wird.
Fluid/Static Layout:	Als Grundlagen wird das Fluid Layout gewählt, umso spätere Anpassungen, wie das Hinzufügen von Elementen und bereitstellen für verschiedene Bildschirmgrößen zu erleichtern.
Grid-Struktur:	Jeder Pagebereich wird in ein Grid-Layout definiert. Dies schafft den Vorteil, dass Elemente übersichtlich strukturiert und im späteren gebrauch effizienter zu ändern sind. So besteht, die in Abbildung 21 zu sehende Liste aller Prozesse in der Hub-Page aus einer Grid-Struktur.

II.b Navigation

Splashscreen:	Als Splashscreen wird eine skalierte Version des Analyse Framework Logos verwendet, dies dient zur besseren Erkennung der App.
Hub-Page:	Die Hub-Page bildet den Startpunkt der App und enthält Informationen über die das Analyse Framework, so wie eine Liste der letzten Prozesse. Wie in Abbildung 21 zu sehen ist, erreicht man durch die Auswahl eines Prozesses auf der Hub-Page eine Section-Page mit den Aktivitäten des gewählten Prozesses.
Section-Page:	Jede Section-Page beinhaltet eine Liste aller Aktivitäten die zu einem Prozess gehören.
Detail-Page:	Eine Auswertungen einer Aktivität und das Erstellen solcher, werden im Detail-Page Template dargestellt.
[Optional] App bars	Eine App bar findet hier keine Verwendung, da sich die Möglichkeiten an ver-

	schiedenen Optionen stark begrenzt.
[Optional] Charms	Charms werden nicht Verwendet, da es nicht nötig ist Menu-Optionen zu verstecken, da diese in einem überschaubaren Rahmen liegen und direkt auf der Hub-Page implementiert werden.

II.c Touch

Touch Input:	Input wird über die Standard Tap-Funktion realisiert. Eingaben werden über die Screen Tastatur getätigt. Dies sind die Standard Eingabefunktionen von Microsoft und benötigen für die Konzeption keine weitere Konfiguration, da diese Standartmäßig bereits problemlos funktioniert.
Touch Feedback:	Feedback erhält der User über Flyouts. Diese sind bereits optimal in das Windows 8 Design eingebaut und können bei einer Feedback Benachrichtigung ohne große Konfiguration verwendet werden.
Inhalte durch Touch:	Scrollen durch die Inhalte wird in einer Horizontalen Swipe-Bewegung realisiert. Diese Touch-Geste ist bereits Standardmäßig in Microsoft enthalten und bedarf keiner weiteren Konfiguration.
Menus durch Touch:	Für das erste Konzept sind keine Menus, wie App-bar oder Charms, vorgesehen. Somit sind alle Optionen auf dem Bildschirm sichtbar und direkt verwendbar. Sollten bei Erweiterungen weitere Menu Optionen hinzugefügt werden, wäre es Sinnvoller diese in einer App-bar oder einem Charm-Menu zu strukturieren um eine Konsistente App-Struktur und Navigation zu erhalten.

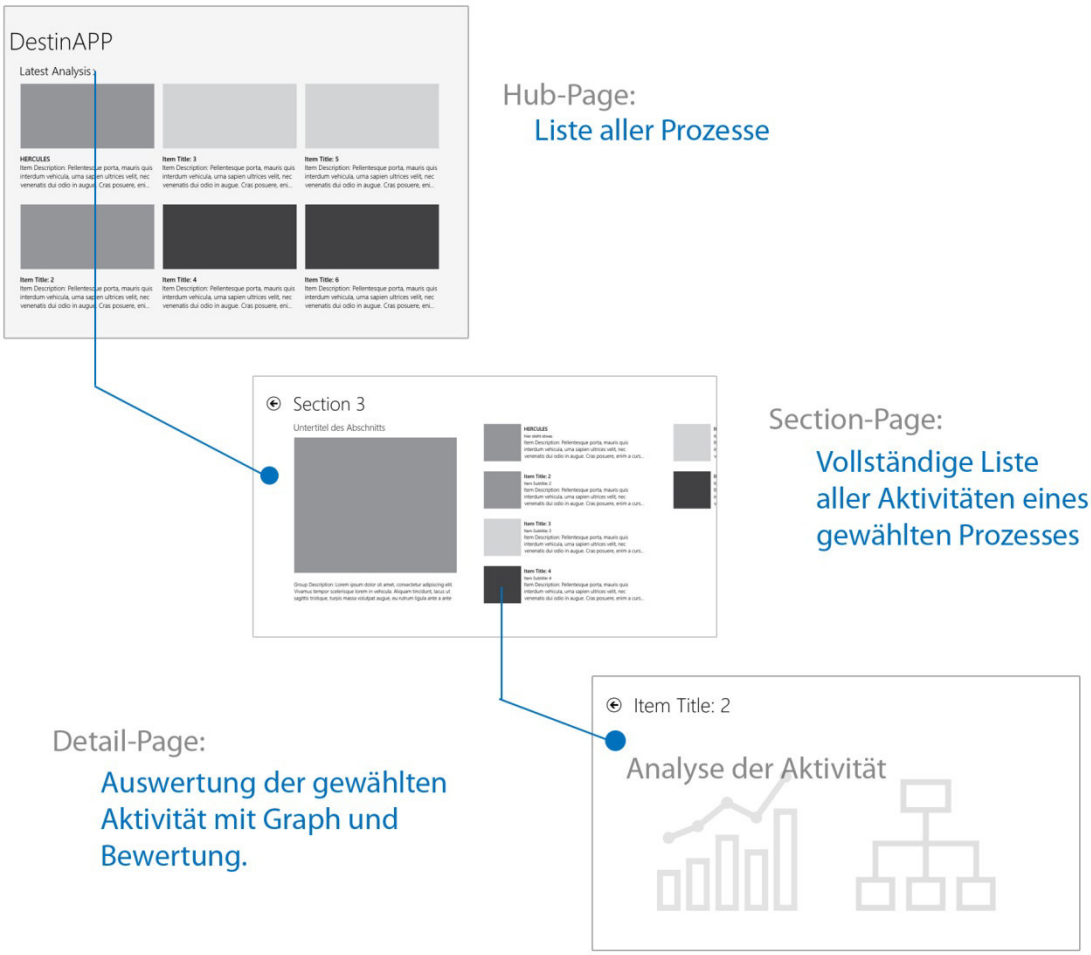

Abbildung 21 – DestinAPP Screenshots: Navigation von Prozess bis zur Auswertung einer Aktivität.

II.d Animationen und Notifikationen

Context menus:	Es werden keine Context menus verwendet. Diese Funktionen für die DestinAPP belanglos und
Flyouts:	Flyouts werden bei besonderen Fehlersituationen wie falschen Eingaben und zur Bestätigung von Eingaben angezeigt. So erhält der User Feedback über eingaben oder Reaktionen der DestinAPP.
Toasts:	Es werden keine Toasts verwendet, da die App keine Notifikationen an den User übermitteln muss.

III Universal App

Erstellen der Universal App:	Projektordner wurde wie in Abbildung 19 zu sehen ist als Universal App erstellt. Des Weiteren wurde der Name „DestinAPP“ als Projektname vergeben.
Definieren gemeinsame Languagefiles:	Eine Language Datei für die deutsche Sprache wurde in DestinAPP.shared erstellt.
Definieren gemeinsamer Daten:	Schnittstelle für gemeinsame Daten wurde in DestinAPP.shared.DataModel erstellt.
Definieren gemeinsamer Assets:	Gemeinsame Bilder werden im Ordner DestinAPP.shared.DataModel erstellt.
Erstellen des Design für Windows 8 App:	Views und Pages wurden in DestinAPP.windows erstellt.
Erstellen des Design für Windows Phone App:	Views und Pages wurden in DestinAPP.phone erstellt.
[Optional] Klassen oder Datenstrukturen für einzelne Plattformen:	Vorerst Datenstrukturen nur für die Windows App erstellt. Für weitere Ausbauten auch für Phone möglich.
[Optional] Ansprechen von Sensoren für Phone oder Tablet:	Es werden keine Sensoren angesprochen.

V Onlinestellen der App

Die App soll nur für den internen Gebrauch zur Verfügung stehen. Somit folgt das Deployment nicht über den App-Store. Eine Zertifizierung der App wäre um jeweilige Fehler zu beseitigen und eine stabil laufende App zu garantieren trotzdem nötig.

8 Diskussion

Hinsichtlich des Kriterienkatalogs und dessen Nutzung als Checkliste, wird dieser in diesem Abschnitt kritisch betrachtet, wobei auf seine Vorteile und eventuell auftretenden Komplikationen und Nachteile eingegangen wird. Dabei wurde der Kriterienkatalog anhand der Entwicklung eines Prototypen, mit dem Namen DestinAPP, getestet und wird nun Evaluiert.

In Anbetracht der neuen Struktur von Windows 8 und des im Grunde unterschiedlichen Entwicklungsprozesses von nativen Windows Desktop Programmen bietet der Kriterienkatalog ein gutes Verfahren, von der Vorauswahl bis hin zum abschließenden online stellen einer App, für den Einstieg in die Windows 8 App Entwicklung.

Den Grundbaustein bilden hierbei die Rahmenbedingungen. Beginnend bei der allgemeinen Gerätekompatibilität bis hin zur Entwicklungssprache, zeigten die Punkte alle wichtigen Aspekte auf um schnell die Entwicklungsumgebung für DestinAPP zu erstellen. In diesem Schrittweise ablaufenden Prozess sind alle Punkte, bis auf die Wahl der Programmiersprache, eindeutig und erfordern keinerlei weitere Ausführung da sie lediglich nacheinander ausgeführt werden müssen oder bei bereits erfülltem Kriterium schnell abgehakt werden können. Auch kann bei Fragen der Grundlagenteil dieser Bachelorarbeit als Hilfe genutzt werden. Somit werden wichtige Schritte nicht vergessen und die Entwicklung kann zügig voran schreiten. Eine Schwachstelle weist dieser Abschnitt des Kriterienkatalogs im Punkt „Wahl der Programmiersprache“ auf, da dieser zu theoretischer Orientierungslosigkeit führt und einen sehr schwammigen Oberbegriff bietet. An diesem Punkt ist der Nutzer des Kriterienkatalogs gezwungen sich tiefer mit den unterschiedlichen Sprachen zu beschäftigen, sofern er keine der vier möglichen Programmiersprachen beherrscht, um nach Abwägung Ihrer Feinheiten eine optimale Wahl für sich zu treffen. Das Aufführen einer komplexen Entscheidungshilfe zur Wahl einer passenden Sprache würde eine einfache und schnelle Handhabung verhindern.

Mit dem darauf folgenden Abschnitt „Design“, liefert der Katalog wesentliche Aspekte zur Gestaltung und Strukturierung einer App. Diese können als Grundlagen der Patterns

betrachtet werden und bieten einen schnellen Überblick, in die wichtigsten Bereiche der App-Gestaltung. Allerdings können hier keine festen Aussagen getroffen werden sondern nur die Grundzüge und Patterns an den Entwickler heran getragen werden, da diesem die künstlerische Freiheit zugesprochen werden muss. Einem Neueinsteiger in die Entwicklung von Windows 8 Apps hilft dieser Bereich allerdings insofern, indem ihm wichtige Aspekte schnell und in Kurzfassung bereit gestellt werden und bei einer möglichen Entscheidung ein Kriterium zu nutzen, kann der Entwickler dies in den User Experience Guidelines von Microsoft schnell und direkt finden.

Der nächste Abschnitt beinhaltet die Kriterien zur Entwicklung mittels einer Universal App. Dieses Verfahren kann besonders dafür genutzt werden, eine breite Masse an User zu erreichen, indem man seine App sowohl für Smartphones als auch Desktop und Tablet anbietet. Der Kriterienkatalog bietet an dieser Stelle, eine gute Schritt-für-Schritt Lösung um schnell ein Projekt zu erstellen und einen Überblick über die Projektstruktur einer Universal App. Allerdings ist ein einlesen in eine tiefergehende Programmierung unumgänglich, da die Optionen nicht zu verallgemeinern wären die von Projekt zu Projekt auftreten könnten.

Der letzte Abschnitt des Kriterienkatalogs beschäftigt mit dem Onlinestellen einer App. Auch in diesem Punkt bildet der Kriterienkatalog eine sehr gute Grundlage um ein Verständnis für die Thematik zu erhalten. Allerdings fehlen auch hier in einigen Schritten Details die für das Onlinestellen einer App nötig sind.

Abschließend ist der Kriterienkatalog als Leitfaden und als Checkliste für Einsteiger in die Windows 8 App Entwicklung ein guter Einstieg und kann genutzt werden um schnell die Groben Aspekte der Entwicklung abzuschließen und wichtige Schrittfolgen einzuhalten.

Für tiefergehendes Verständnis einzelner Kriterien muss der Entwickler die Entwicklerseite von Windows 8 oder sonstige Literatur zur Hilfe nehmen. Die Weiterführung des Kriterienkatalogs oder die Ausführung bestimmter Kriterien auf eine tiefergehende Abstraktionsebene würde die Funktion des Kriterienkatalogs als „schnelle Checkliste“ sprengen und würde den Aufwand um einiges erhöhen damit zu arbeiten.

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich diese Arbeit selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht.

Diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

_____

Ort, Datum, Unterschrift

Literaturverzeichnis

[BN09] Bevan, Nigel. "What is the difference between the purpose of usability and user experience evaluation methods." *Proceedings of the Workshop UXEM*. Vol. 9. 2009.

[JH06] Jochen Ludewig, Horst Lichter. „Software Engineering“ Vol. 2. 2006.

[GJC85] Gould, John D., and Clayton Lewis. "Designing for usability: key principles and what designers think." *Communications of the ACM* 28.3 (1985): 300-311.

[SB13] Schooley, Brent. *Designing for Windows 8*. Apress, 2013.

[HGKM] Hoos Eva, Gröger Christoph, Kramer Stefphan, Mitschang Bernhard „Improving Business Processes through Mobile Apps - An Analysis Framework to Identify Value-added App Usage Scenarios.” Proceedings of the 16th International Conference on Enterprise Information Systems (ICEIS), 27-30 April, 2014, Lisbon, Portugal:

[ISO] International Organization for Standardization (ISO) “Part 210: Human-centred design for interactive systems” URL(24. Oktober 2014):
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=52075

[MNum] Microsoft “Microsoft by the Numbers”
URL(24. Oktober 2014):
http://www.microsoft.com/en-us/news/bythenumbers/index.html

[MZert] Microsoft „Tests des Zertifizierungskits für Windows-Apps für Windows Store-Apps. Unter Mitarbeit von Microsoft Dev Center. Hg.v. Microsoft.“ URL(24. Oktober 2014):
http://msdn.microsoft.com/de-de/library/windows/apps/jj657973.aspx#crashes_and_hangs_test

[MUX] Microsoft „User experience guidelines: Windows 8 User experience guidelines 2012“
URL(24. Oktober 2014):
http://msdn.microsoft.com/de-de/library/windows/apps/hh465424.aspx

[NET] NetMarketShare „Desktop Top Operating System Share Trend“ 2014
URL(24. Oktober 2014):
http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=11&qpcustomb=0

[PCW] PC-Welt „Windows 8: Hybrides Booten (Fast Boot) – Windows 8 Funktionen“
URL(24. Oktober 2014):
http://www.pcwelt.de/ratgeber/Windows-8-Funktionen-Hybrides-Booten-Fast-Boot-6632384.html