bbv
MAKING VISIONS WORK.

BOOKLET

MOBILE: ENTWICKLUNG, UX & TESTING

PROFITIEREN SIE VON UNSERER ERFAHRUNG!

Kontakt Schweiz
bbv Software Services AG
Blumenrain 10
6002 Luzern
Telefon: +41 41 429 01 11
E-Mail: info@bbv.ch

Kontakt Deutschland
bbv Software Services GmbH
Agnes-Pockels-Bogen 1
80992 München
Telefon: +49 89 452 438 30
E-Mail: info@bbv.eu

INHALT

1 MOBILE IST ÜBERALL

Montagmorgen um 7:30 am Hauptbahnhof in Zürich. Hunderte Menschen auf dem Weg zur Arbeit. Ein vorbeieilender Geschäftsmann im Anzug prüft soeben die neuesten E-Mails auf seiner Smartwatch. Der jugendliche Pendler neben uns vertreibt sich die Wartezeit beim Spielen auf seinem Handy, sein Kollege prüft via Streamingdienste die neuesten Songs. Vor uns am Bahnsteig liest eine ältere Frau wie gebannt einen Roman auf ihrem E-Book-Reader. Es ist offensichtlich: Neben Kaffee und Gratiszeitung sind Mobilgeräte jeder Art die wichtigsten Begleiter.

Die Allgegenwärtigkeit von Mobilgeräten wird durch verschiedene Studien bestätigt. Gemäss Media Use Index[1] aus dem Jahr 2016 greifen 91% der Schweizer mit Mobilgeräten auf das Internet zu. Dieselbe Befragung hat ferner ergeben, dass ein beachtlicher Anteil der Bevölkerung ihre Mobilgeräte im Schlafzimmer und sogar auf der Toilette mit dabeihaben!

Andere Untersuchungen fördern zum Teil skurrile Fakten ans Tageslicht. So hat eine Datenanalyse[2] aus dem Jahr 2011 gezeigt, dass weltweit etwa 4.8 Milliarden Mobiltelefone existieren, aber nur 4.2 Milliarden Zahnbürsten. Oder anders formuliert: Mobiltelefone scheinen wichtiger als Zahnbürsten zu sein.

1.1 CHANCEN UND GEFAHREN

Die steigende Verbreitung der Mobilgeräte hat – zusammen mit Technologien wie Cloud oder Internet of Things – auch grossen Einfluss auf die Wirtschaft. Nehmen wir als Beispiel den Fahrgast-Vermittlungsdienst Uber[3]. Gemäss Analysten betrug der Umsatz mit Fahrtenbuchungen im Jahr 2015 ungefähr 11 Milliarden Dollar, wovon 20% als Vermittlungsgebühr an Uber gehen[4]. Uber macht somit – knapp 6 Jahre nach der Firmengründung – mehr als 2 Milliarden Umsatz. Und das Wall Street Journal schätzte den Wert der Firma kürzlich auf über 50 Milliarden Dollar ein[5]!

[1] http://www.media-use-index.ch/
[2] http://60secondmarketer.com/blog/2011/10/18/more-mobile-phones-than-toothbrushes/
[3] https://www.uber.com/de/
[4] https://de.wikipedia.org/wiki/Uber_(Unternehmen)
[5] https://www.welt.de/wirtschaft/webwelt/article145258717/Warum-ist-Uber-so-viel-wert.html

Der grosse Erfolg von Uber liegt darin, dass er die technologischen Fortschritte genutzt hat, um eine bisher nie da gewesene, neue Dienstleistung zu schaffen. Fahrgäste können sich dank der Uber-App schneller, bequemer und günstiger als jemals zuvor transportieren lassen. Durch eine simple Idee bringt die Firma etablierte Transportunternehmen auf der ganzen Welt in arge Bedrängnis.

Neben der Möglichkeit für völlig neue Dienstleistungen bieten Apps weitere Chancen:

- Funktionen wie Push-Notifikationen oder «Live Support» schaffen neue Kommunikationskanäle und können die Kundenbindung verbessern.
- Über mobile Lösungen können neue Kundensegmente erschlossen werden, die bis anhin vielleicht kein Thema waren.
- Durch Digitalisierung und Optimierung bestehender Prozesse können oft Kosten eingespart werden.

Es gibt also gute Gründe, sich mit den Möglichkeiten von Mobilgeräten bzw. mobilen Applikationen zu beschäftigen. Leider wird aber oft die Komplexität der Thematik stark unterschätzt. Aus diesem Grund haben wir das vorliegende Booklet geschaffen. Es soll einen Einblick in die Herausforderungen der App-Entwicklung bieten – zusammen mit geeigneten Lösungsansätzen.

2 MOBILE IST MEHR ALS SMARTPHONES

Wenn von «Mobile» gesprochen wird, so denken die meisten Menschen zuerst einmal an Smartphones. Das Themengebiet umfasst aber viel mehr. Dieses Kapitel liefert zur Horizonterweiterung einen Überblick über die momentan wichtigsten Gerätetypen, Sensoren und Aktoren sowie Betriebssysteme.

2.1 GERÄTETYPEN

Die momentan wichtigsten Gerätetypen am Markt sind:

- Feature Phone
- Smartphone
- Smartwatch
- Tablet

Daneben existieren viele weitere, für spezielle Anwendungsfälle optimierte Gerätetypen. Meist werden diese Geräte mit leicht modifizierten Versionen der Smartphone-Betriebssysteme betrieben:

- Phablets
- Convertibles
- E-Book-Reader
- Spielkonsolen
- Embedded-Geräte
- Fitness-Armbänder
- Smart TV
- Car-Infotainmentsysteme
- Augmented Reality
- Virtual Reality

2.2 SENSOREN UND AKTOREN

Moderne Mobilgeräte bieten eine Vielzahl an Sensoren und Aktoren. Darunter finden sich auch immer mal wieder skurrile Dinge, beispielsweise Smartphones mit Wärmebildkameras[6] oder Iris-Scannern[7]. Durch sinnvolle Integration dieser Möglichkeiten in die eigenen Applikationen können neue und spannende Nutzungsmodelle und Dienstleistungen geschaffen werden. Eine nicht abschliessende Sammlung an verfügbaren Sensoren und Aktoren:

Eingabe (1)	Eingabe (2)	Ausgabe	Kommunikation
• Touchscreen	• Elektromagnetisch	• Display	• GSM
• Keyboard	• Barometer	• E-Ink Display	• GPRS/EDGE
• Mikrofon	• Thermometer	• LED	• UMTS
• Kamera	• Magnetometer	• Lautsprecher	• LTE
• GPS	• Fingerabdruck	• Vibration	• Infrarot
• Beschleunigung	• Pulsmesser	• Haptisch	• Bluetooth
• Lage	• Iris-Scanner		• WLAN
• Näherung	• Gasdetektor		• LAN
• Helligkeit	• Wärmebildkamera		• NFC/RFID

2.3 BETRIEBSSYSTEME

Die beiden grossen Betriebssysteme im Mobile-Bereich sind momentan Apple iOS[8] und Google Android[9]. Weit abgeschlagen auf dem dritten Platz befindet sich Microsoft Windows Mobile[10]. Regional gibt es grosse Unterschiede in den Marktanteilen (Tabelle 1).

[6] http://www.heise.de/newsticker/meldung/Caterpillar-S60-Robustes-Smartphone-mit-Waermebildkamera-3111856.html
[7] http://www.golem.de/news/microsoft-lumia-950-xl-im-test-schau-mir-in-die-augen-windows-1512-117827.html
[8] http://www.apple.com/chde/ios
[9] https://www.android.com/
[10] https://www.microsoft.com/de-ch/mobile/windows10/

Die Schweiz ist beispielsweise ein sehr starkes Land für Apple. Wesentlicher Hauptgrund hierfür ist die hohe Zahlungskraft der schweizerischen Bevölkerung.

Tabelle 1

	Weltweit[11]	Europa[12]	Schweiz[13]
Android	87%	69%	42%
iOS	12%	26%	54%
Windows Mobile	<1%	3%	3%
Andere	<1%	2%	1%

Schauen wir uns nachfolgend die drei wichtigsten Betriebssysteme mit ihren Eigenschaften etwas genauer an.

Abbildung 1:
Logo von Android

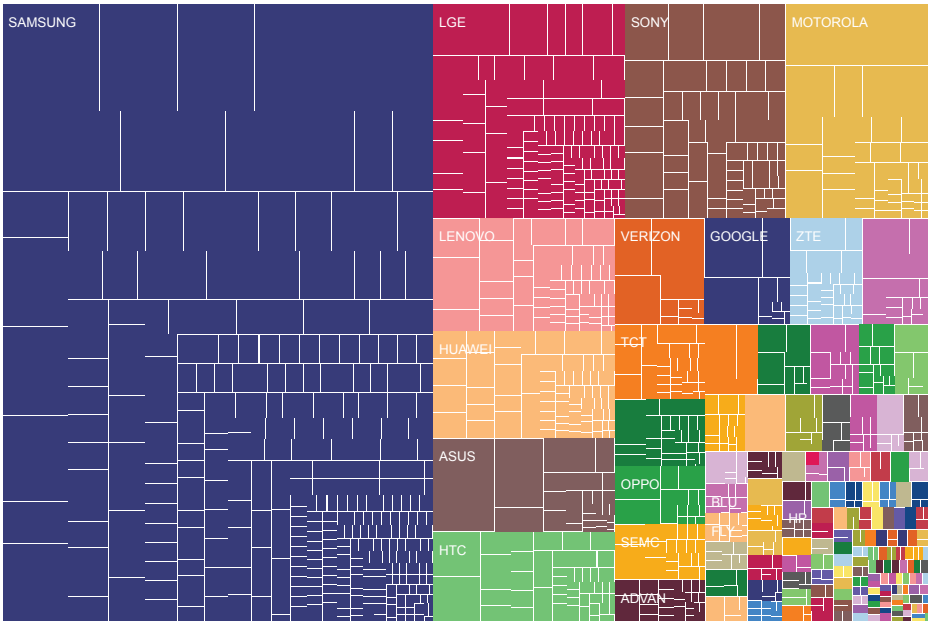
Google Android

Android wird von der von Google gegründeten Open Handset Alliance (OHA) entwickelt. Die OHA ist ein Konsortium von 84 Unternehmen, das die Schaffung von offenen Standards für Mobilgeräte als Ziel hat. Android ist freie, quelloffene Software und basiert auf dem Linux-Kernel. Varianten vom «normalen» Android für Smartphones und Tablets gibt es für Smartwatches (Android Wear) und Autos (Android Auto).

[11] http://www.idc.com/prodserv/smartphone-os-market-share.jsp
[12] https://de.statista.com/statistik/daten/studie/184516/umfrage/marktanteil-der-mobilen-betriebssysteme-in-europa-seit-2009/
[13] http://www.computerworld.ch/marktanalysen/studien/artikel/apple-bleibt-marktfuehrer-in-der-schweiz-android-holt-auf-69694/

Aufgrund der freien Verfügbarkeit von Android existiert eine riesige Zahl an unterschiedlichen Android-Geräten auf dem Markt. Man spricht in diesem Zusammenhang auch von «Fragmentierung». Die Hersteller der OpenSignal-App haben im August 2015 die Downloads ihrer App im Google Play Store analysiert[14]. Die 682 000 untersuchten Downloads haben sich auf, sage und schreibe, 24 093 unterschiedliche Gerätekonfigurationen verteilt.

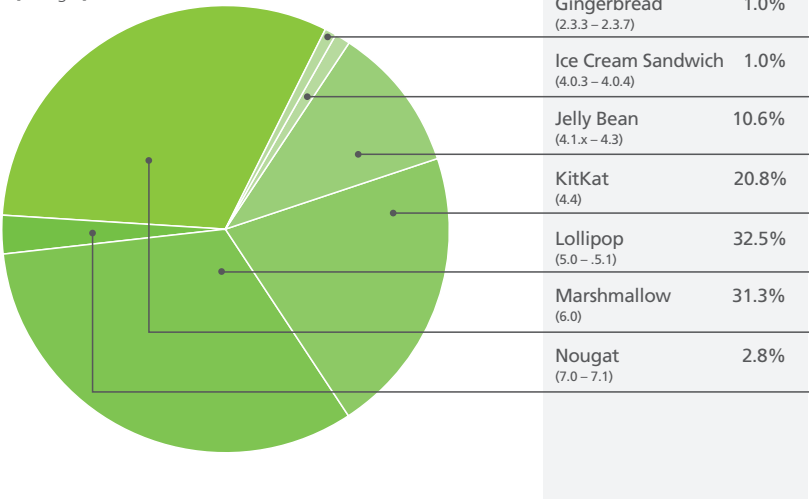
Abbildung 2: Die 24 093 Geräte-Typen grafisch dargestellt [OpenSignal]



[14] https://opensignal.com/reports/2015/08/android-fragmentation/

Das Problem der Fragmentierung wird durch den Umstand weiter verschärft, dass viele der Hersteller ihren Android-Geräten eine angepasste grafische Oberfläche verpassen. Da die Pflege der grafischen Oberfläche zeit- und kostenintensiv ist, erhalten gerade günstigere Geräte oft keine Updates auf neue Android-Versionen. So lässt sich auch erklären, warum das mittlerweile über drei Jahre alte Android 4.4 (KitKat) noch einen Marktanteil von fast 21% für sich beanspruchen kann[15].

Abbildung 3: Verteilung der Android-Versionen im Markt [Google]



[15] https://developer.android.com/about/dashboards/index.html

Apple iOS

iOS ist ein von Apple entwickeltes mobiles Betriebssystem für das iPhone, das iPad und den iPod. Auf iOS basieren ausserdem watchOS für die Apple Watch sowie tvOS für das Apple TV. iOS baut auf dem OS-X-Kern von Apple auf.

Abbildung 4:
Logo von Apple iOS

iOS wird nur auf eigener Hardware von Apple eingesetzt – im Gegensatz zu Android und Windows Mobile. Dadurch ist die Vielfalt an Geräten und Betriebssystemversionen im Markt viel geringer als bei Android. Dies erleichtert die Aktualisierung der Geräte erheblich. Als beispielsweise im Herbst 2015 iOS 9 auf den Markt kam, war dieses nach sechs Tagen schon auf über 50% aller Apple-Geräte installiert[16].

Die Geschlossenheit der Plattform ist zugleich aber auch der grosse Nachteil der Plattform – Geräte von alternativen Herstellern sucht man vergebens. Dasselbe gilt für die Entwicklungstools: Entweder man nimmt, was einem Apple liefert, oder man verzichtet auf die Plattform. Dieser «Zwang» geht sogar so weit, dass ein Apple Computer mit macOS als Entwicklungsgerät benötigt wird.

Microsoft Windows Mobile

Die Mobile-Sparte gilt als grosses Sorgenkind bei Microsoft. Nach zahlreichen Umbenennungen – die Plattform wurde mehrmals neu konzipiert – ist man mit Windows 10 nun wieder zum Namen Windows Mobile zurückgekehrt.

Abbildung 5:
Logo Windows 10 Mobile

Windows 10 Mobile basiert auf demselben Windows-NT-Kernel wie das Betriebssystem Windows 10. Das System wurde für Geräte

[16] http://www.golem.de/news/apple-ios-9-soll-bisher-schnellste-update-rate-aufweisen-1509-116421.html

mit ARM-Prozessor mit einer Bildschirmdiagonale bis 8 Zoll entwickelt. Geräte mit grösseren Bildschirmen (z. B. Tablets) werden mit der Desktop-Variante von Windows 10 ausgeliefert. Aufgrund des gemeinsamen Betriebssystemkerns sind Anwendungen für verschiedene Gerätetypen vergleichsweise einfach umzusetzen.

Abbildung 6: Windows 10 – Universal Windows Platform

Im deutschsprachigen Raum gibt es momentan nur drei Geräte mit Windows 10 Mobile, allesamt von Microsoft[17]. Daran wird sich so schnell auch nichts ändern: Im Sommer 2016 hat Microsoft bekannt gegeben, dass sie selber keine Hardware mehr produzieren werden[18]. Ob und in welcher Form Windows Mobile weiterentwickelt wird, ist offen. Die Wahrscheinlichkeit, dass andere Hersteller auf die Plattform aufspringen werden, ist aufgrund fehlender Nachfrage am Markt mehr als gering.

[17] https://de.wikipedia.org/wiki/Microsoft_Windows_10_Mobile#Verf.C3.BCgbarkeit_und_Marktanteile
[18] https://www.mobilegeeks.de/artikel/kommentar-lumia-ist-tot-es-lebe-windows-10-mobile/

3 HERAUSFORDERUNGEN

Die Entwicklung qualitativ hochwertiger Mobilapplikationen ist alles andere als einfach. Grund genug, die wichtigsten Problemfelder etwas genauer zu betrachten.

3.1 VIELFALT

Lassen Sie uns zu Beginn ein kurzes Gedankenspiel machen: Wie viele Smartphone- und Tablet-Hersteller kennen Sie? Wie viele unterschiedliche Modelle haben diese Hersteller im Angebot? Und wie viele unterschiedliche Betriebssystem-Versionen laufen wohl auf all diesen Modellen?

Vermutlich haben Sie in Ihrem Kopf soeben eine ziemlich grosse Zahl ermittelt. In genau dieser grossen Zahl liegt die grösste Herausforderung der App-Entwicklung: die kombinatorische Vielfalt. Klassische Desktop-Software muss meist nur auf einer Plattform laufen. Mobile Apps, die unter Umständen von Millionen Menschen auf der ganzen Welt verwendet werden, hingegen auf den unterschiedlichsten Plattformen. Die drei wichtigsten veränderlichen Einflussfaktoren auf den Entwicklungsaufwand sind:

Gerätetypen

Auf welchen Gerätetypen soll die App laufen? Genügt das Smartphone oder sollen auch Tablets unterstützt werden? Eventuell sogar Smartwatches? Grundsätzlich gilt: Je mehr Arten unterstützt werden sollen, desto grösser der Entwicklungsaufwand.

Hardware

Die Hardware der Mobilgeräte unterscheidet sich in diversen Punkten, beispielsweise: CPU, Memory, Displaygrösse, Sensoren und Aktoren. Erschwerend kommt hinzu, dass die verbauten Teile von unterschiedlicher Qualität sind. Ferner kann der Anwender der App den Zugriff auf Hardware auch untersagen. All diese Aspekte haben wiederum Einfluss auf den Aufwand zur Umsetzung der App.

Betriebssysteme

Die Welt der Desktop-Software ist vergleichsweise einfach: Trotz wachsendem Marktanteil von Apple läuft noch immer auf fast jedem Heimcomputer Microsoft Windows. Nicht so bei Mobilgeräten: Google Android und Apple iOS sind in der Schweiz die beiden dominierenden Betriebssysteme. Sollen möglichst viele Endbenutzer erreicht werden, so muss eine App mindestens für diese beiden Betriebssysteme angeboten werden, was wiederum Mehraufwand bedeutet.

3.2 VERÄNDERUNG

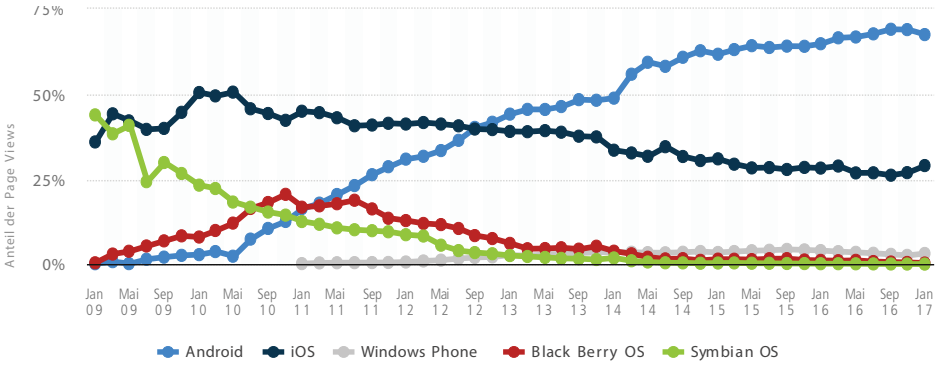
Können Sie sich an BlackBerry-Geräte erinnern? Oder an das Betriebssystem Symbian OS? Beide konnten noch vor wenigen Jahren beachtliche zweistellige Anteile am Markt für sich beanspruchen. Und beide sind heute so gut wie von der Bildfläche verschwunden, siehe auch Abbildung 7.

Dies bringt uns zur zweiten grossen Herausforderung: Dem steten Wandel im Mobile-Markt. Produkte, die heute extrem gefragt sind, könnten in wenigen Jahren Auslaufmodelle sein. Wir sollten also Lösungen entwickeln, die möglichst unabhängig von der darunterliegenden Hard- und Software sind. So können wir unsere Anwendungen bei Bedarf auf neue Plattformen migrieren.

Die hohen Änderungsraten haben auch unmittelbare Auswirkungen: Apple, Google und Microsoft geben im Abstand weniger Monate neue Versionen ihrer Betriebssysteme heraus. Dies bedeutet wiederum, dass Apps für neue Versionen optimiert und getestet werden müssen. Und auch «hinter den Kulissen» ist steter Wandel an der Tagesordnung: Wissen über neue Program-

miersprachen, Tools und Konzepte muss erarbeitet und angewendet werden. Um in diesem schnelllebigen Umfeld bestehen zu können, braucht es hochqualifiziertes und lernbegieriges Personal.

Abbildung 7: Verbreitung der Betriebssysteme zwischen 01/2009 und 09/2016[19]



3.3 HARDWARE

Im Unterkapitel 3.1 haben wir den Punkt «Hardware» unter dem Aspekt der Vielfalt beleuchtet. Das Thema bietet aber noch weitere Herausforderungen:

Rechenleistung

Im Gegensatz zu Computern besitzen Mobilgeräte weniger Leistung und Speicher. Aus diesem Grund müssen Apps ressourcenschonend und hoch performant implementiert werden.

[19] https://de.statista.com/statistik/daten/studie/184516/umfrage/marktanteil-der-mobilen-betriebssysteme-in-europa-seit-2009/

Verfügbarkeit

Die verfügbare Hardware kann von Gerät zu Gerät sehr unterschiedlich sein. Das gilt insbesondere für Sensoren und Aktoren. Bei modularen Smartphones, wie beispielsweise dem mittlerweile eingestellten Google Ara[20], kann sich die Hardwareausstattung sogar während des Betriebs ändern. All diese Fälle müssen Apps erkennen und fehlerfrei behandeln können.

Abbildung 8:
Modulares Smartphone
«Project Ara»

Testbarkeit

Das Testen von Hardware-Funktionalität ist eine anspruchsvolle Aufgabe. Wie kann man prüfen, dass eine App bei fehlendem Signalempfang weiterhin funktioniert? Wie kann die Korrektheit einer Kamera- oder Vibrationsfunktionalität geprüft werden? Und wie können solche Tests zwecks Wiederholbarkeit automatisiert werden?

[20] https://atap.google.com/ara/

3.4 SOFTWARE

Technologien

An früherer Stelle wurde bereits betont, dass die Vielfalt und Änderungsrate bei Betriebssystemen, Tools und Konzepten extrem hoch ist. Die diversen Möglichkeiten bieten unterschiedliche Vor- und Nachteile, die je nach Anforderungen gegeneinander abgewogen werden müssen. Die Wahl der richtigen Technologie(n) ist mitunter die schwierigste Entscheidung zu Beginn der Entwicklung.

Spezialfälle

Mobile-Apps müssen mit Situationen umgehen können, die bei Desktop-Anwendungen kein Thema sind. Wie verhält sich die App bei einem Ausfall der Internetverbindung? Wie reagiert die Anwendung bei einem eingehenden Telefonanruf? Was passiert, wenn das Betriebssystem die App in den Hintergrund stellt und ihr die CPU entzieht?

Bedienkonzepte

Kleinere Displays und andere Ein- und Ausgabegeräte verlangen nach angemessenen Bedienkonzepten. Die vom PC gewohnten Muster können ohne geeignete Adaptierung nicht auf Mobilgeräte übernommen werden.

User Interface Guidelines

Alle grossen Betriebssystem-Hersteller besitzen Vorgaben und Leitlinien für den Entwurf der grafischen Oberfläche. Die darin enthaltenen Konzepte sind teilweise gleich, haben aber auch grosse Unterschiede. Das Kennen und korrekte Anwenden der Leitlinien ist wichtig für die Zulassung in den offiziellen App Stores und für die Akzeptanz bei den Endbenutzern.

3.5 WEITERES

Erwartungshaltung

Endbenutzer haben sehr hohe Erwartungen an die Qualität einer App. Werden die Erwartungen nicht oder nur teilweise erfüllt, so äussert sich das rasch in schlechten Bewertungen im App Store. Dies wiederum kann dem Ansehen der Unternehmung oder Marke empfindlich schaden.

Preisdruck

Hand aufs Herz: Wann haben Sie zum letzten Mal für eine App in einem der App Stores bezahlt? Und wie hoch war der Betrag? Studien zeigen, dass sich viele Endbenutzer mit Gratis-Apps zufriedengeben. Nur in seltenen Fällen wird für eine App bezahlt. Es stellt sich also früh in der Entwicklung die Frage, wie eine App gewinnbringend entwickelt werden kann.

Interdisziplinarität

Die Entwicklung von mobilen Lösungen tangiert die unterschiedlichsten Rollen: Requirements Engineers, UX-Designer, Softwareentwickler, Testmanager, Projektleiter und mehr. Die Zusammenarbeit in interdisziplinären Teams ist zwar spannend und lehrreich, kann aber auch ziemlich anspruchsvoll sein.

4 AGILES VORGEHEN

Agile Vorgehensmodelle, wie beispielsweise Scrum, eignen sich hervorragend für die Entwicklung mobiler Lösungen. Dieses Kapitel zeigt anhand einiger Beispiele, warum das so ist.

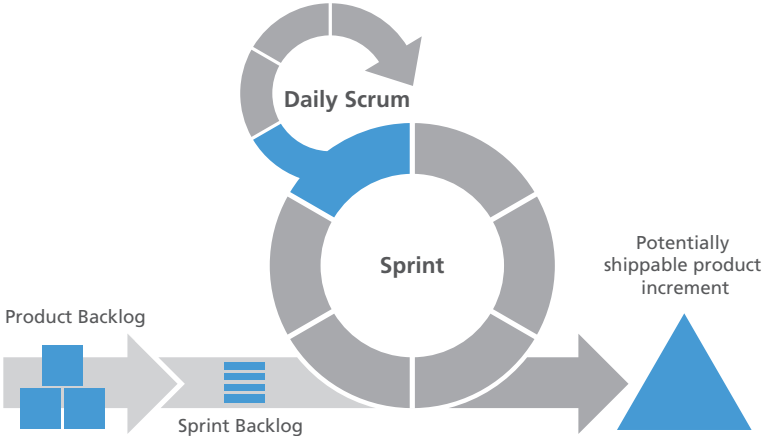


Abbildung 9: Agile Softwareentwicklung nach Scrum[21]

4.1 INTERDISZIPLINÄRE TEAMS

Für die Entwicklung von Mobilapplikationen sind verschiedene Fachexperten nötig. Die Koordination von Experten aus unterschiedlichen Fachbereichen ist nicht immer einfach, weshalb die «Interdisziplinarität» in Kapitel 3 auch als Herausforderung bezeichnet wurde. Agile Vorgehensmodelle basieren auf der Idee, dass selbstorganisierte, interdisziplinäre Teams die besten Lösungen hervorbringen. Insofern liegt es nahe, für die Entwicklung mobiler Lösungen auf ein solches Vorgehen zu setzen.

Damit diese interdisziplinären Teams die erwarteten Ergebnisse auch wirklich produzieren können, sind verschiedene organisatorische Aspekte zu beachten. So ist es beispielsweise von grossem Vorteil, die Teammitglieder räumlich nahe beieinander zu positionieren. Weiter sollten Möglichkeiten vorhanden sein, welche die regelmässige Kommunikation zwischen den einzelnen Teammitgliedern fördern. Beispiele hierfür sind wiederkehrende Status-Meetings (z. B. in Form eines «Standup Meetings») oder die Verfügbar-

[21] http://scrum-master.ch/agile/index.php/menu-main-agile/menu-main-agile-scrum

keit geeigneter Infrastruktur (z. B. Whiteboards für gemeinsame Diskussionen).

4.2 ITERATIV UND INKREMENTELL

Eine Gemeinsamkeit agiler Vorgehensmodelle ist die Idee, iterativ und inkrementell zu der angestrebten Lösung zu kommen. Durch kontinuierliche Überprüfung des Zustands kann bei Abweichungen schnell korrigierend eingegriffen werden. Ausserdem können neue Erkenntnisse und Ideen zeitnah eingebracht werden.

Für die Entwicklung von Mobile-Apps ist ein solches Vorgehen aus unterschiedlichen Gründen vorteilhaft:

- Technische Risiken können früh adressiert werden. Im Idealfall existiert bereits nach der ersten Iteration ein Grundgerüst der Anwendung, das die Machbarkeit aller relevanten Elemente beweist.
- Ein gutes UX-Design braucht Zeit. Die Wahrscheinlichkeit, dass beim ersten Entwurf bereits die «perfekte» Version entsteht, ist sehr klein. Durch das iterative Vorgehen können neue Ideen schnell überprüft und das Design stetig verbessert werden.
- Anforderungen werden in agilen Vorgehensmodellen fortlaufend priorisiert. Dadurch enthält die zu entwickelnde App nur diejenigen Funktionen, die dem Kunden einen wirklichen Mehrwert liefern. Bei Mobile Apps ist das insofern besonders wichtig, da Budgets aufgrund der geringen Preismarge in den App Stores meist sehr knapp bemessen sind. Mit einem «Minimal Viable Product»-Ansatz kann früh eine erste Version in den Stores veröffentlicht werden – wodurch bereits Einnahmen generiert werden. Und ganz nebenbei erhält man durch User Reviews kostenlose Verbesserungsvorschläge geliefert.

5 USER EXPERIENCE (UX)

Nachdem wir uns im Kapitel 3 mit verschiedenen Herausforderungen von mobilen Anwendungen beschäftigt haben, betrachten wir nun Lösungsansätze im Bereich der User Experience.

5.1 BEDEUTUNG

Im App-Dschungel überlebt nur, wer ein einzigartiges Benutzungserlebnis bieten kann. Benutzer erwarten heute eine einfache Bedienung, einen angemessenen Funktionsumfang und ein ansprechendes Design. Bereits beim ersten Start der App muss sich ihr Mehrwert erschliessen. Gelingt dies nicht, wird die App wieder deinstalliert. Eine zweite Chance gibt es meistens nicht. Der Benutzer akzeptiert eine App nur, wenn sie ihn in der Ausführung seiner Ziele optimal unterstützt und irgendetwas in seinem Leben leichter macht.

Abbildung 10: Beispiel eines positiven Benutzungserlebnisses (Touch-Fahrplan der SBB-App). Der Benutzer kann seine wichtigsten Haltestellen als Bild-Kachel definieren und sich mit dem Bewegen des Fingers vom Abfahrts- zum Zielort schnell den gewünschten Fahrplan anzeigen lassen.

Das Benutzungserlebnis entscheidet massgeblich über den Erfolg oder Misserfolg einer App. Einfache und verständliche User Interfaces sorgen zusammen mit selbsterklärenden und leicht erlernbaren Funktionalitäten für positive Bewertungen und eine rasche Verbreitung.

5.2 DEFINITION USER EXPERIENCE

Der Begriff User Experience, abgekürzt «UX», wird mit «Benutzungserlebnis» oder «Nutzererfahrung» übersetzt und umschreibt eine ganzheitliche Sicht auf die Interaktion von Menschen mit der Technik. Technologie soll so gestaltet werden, dass sie effizient, effektiv und auf eine möglichst angenehme Art und Weise bedient werden kann.

User Experience hat zum Ziel, beim Benutzer positive Eindrücke zu erzeugen und ihm das Gefühl zu vermitteln, einen echten Mehrwert zu erhalten. Damit der Benutzer mit der Bedienung der App gut und leicht klarkommt und nachhaltig begeistert wird, müssen verschiedene Disziplinen wie Usability, Informationsarchitektur, Interaktionsdesign, Informationsdesign und Visual Design berücksichtigt werden.

Benefits einer benutzerzentrierten Entwicklung (User-Centered Design, abgekürzt «UCD»):

- einfache & verständliche Bedienung
- schnelle & klare Orientierung
- angemessener Funktionsumfang
- hoher Nutzwert (Gefühl, die App erleichtert das Leben)
- positive Überraschungen («Wow»-Effekt)
- Spass («Joy of use»)
- ansprechendes, ästhetisches Design

Erreicht wird eine positive UX mit einer benutzerzentrierten Entwicklung, welche den Benutzer mit seinen Bedürfnissen und Erwartungen konsequent ins Zentrum stellt.

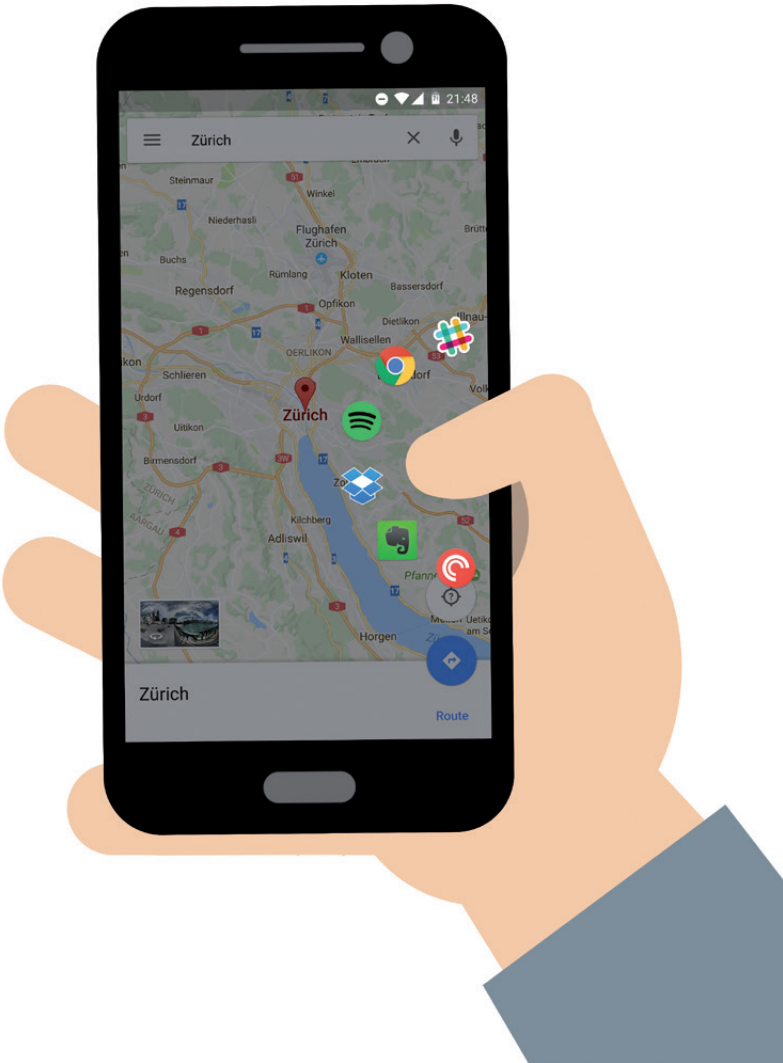


Abbildung 11: Beispiel eines positiven Benutzungserlebnisses (App-Switcher «Swiftly Switch»). Mit einer einfachen Wischgeste vom rechten Seitenrand nach links kann einhändig zu den zuletzt genutzten Apps navigiert werden.

5.3 ANFORDERUNGEN UND ERWARTUNGEN VON BENUTZERN

Ob und welche Emotionen durch ein Produkt beim Benutzer ausgelöst werden, hängt stark mit seinen Anforderungen und Erwartungen zusammen. In einer Benutzeranalyse wird deshalb ein vertieftes Verständnis über die Benutzer mit ihren Fähigkeiten, Rollen und Wertvorstellungen geschaffen. Die Analyse soll klären, welche Vorkenntnisse, Erfahrungen, Einstellungen, Erwartungen und Ziele typische Benutzer haben und welche Akzeptanzkriterien aus Benutzersicht bestehen.

Neben der Erhebung der Benutzeranforderungen werden in der Nutzungskontextanalyse die Merkmale der zu erledigenden Aufgaben und der physischen Umgebung bestimmt. Die Ergebnisse bestimmen, welche Dialog- und Interaktionsmöglichkeiten im Produkt eingesetzt werden können. In dieser Phase wird entschieden, welche Funktionalitäten das neue System benötigt und wie die Aufgaben auf Mensch und Maschine verteilt werden. Anforderungen werden priorisiert, auf ihre technische Machbarkeit überprüft und hinsichtlich ihrer Wirtschaftlichkeit bewertet (Kosten-Nutzen-Analysen).

Um ein positives Benutzungserlebnis zu schaffen, müssen die Erwartungen des Benutzers auf verschiedenen Ebenen erfüllt bzw. übertroffen werden:

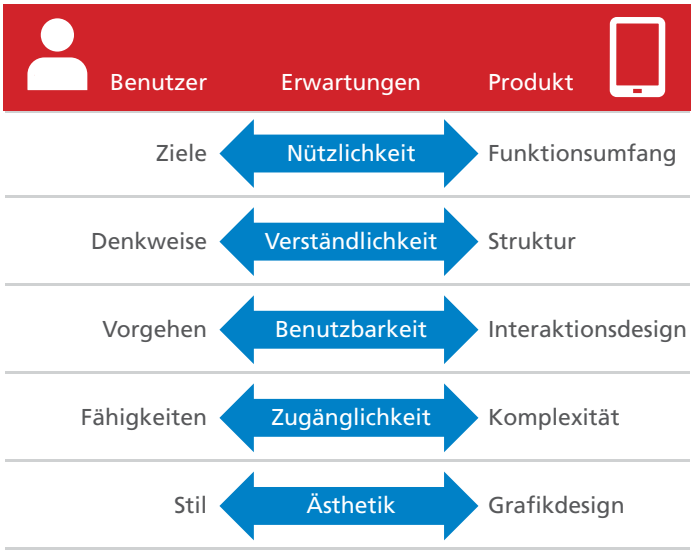


Abbildung 12: Auf verschiedenen Ebenen treffen Eigenschaften des Benutzers auf Merkmale des Produkts. Für eine positive User Experience müssen die Erwartungen auf jeder Ebene erfüllt werden (aus Moser, 2012).

KANO-MODELL

Eine verbreitete Methode zur Klassifikation von Anforderungen ist das Kano-Modell, das Produktmerkmale in Basis-, Leistungs- und Begeisterungsmerkmale einteilt. Das Kano-Modell beschreibt den Zusammenhang zwischen dem Erreichen bestimmter Produktmerkmale und der erwarteten Zufriedenheit des Benutzers (vgl. Moser, 2012).

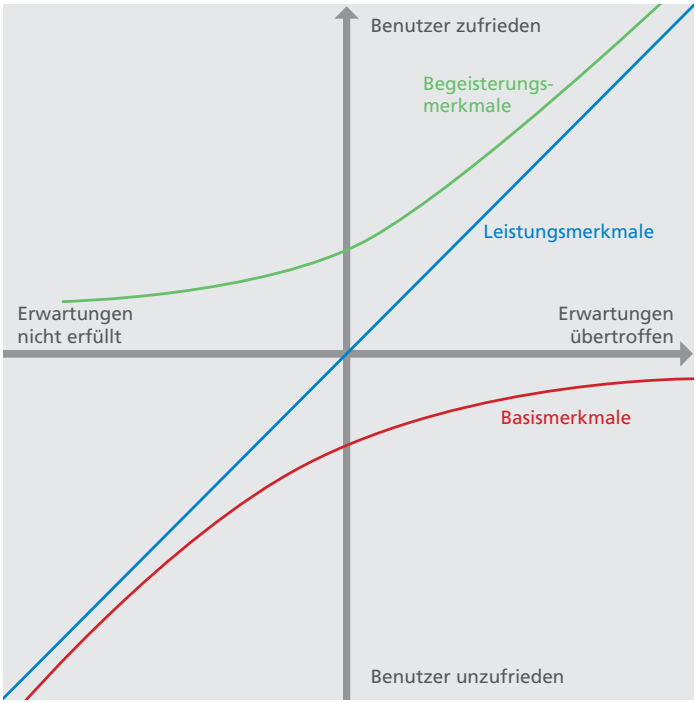


Abbildung 13:
Kano-Modell

Basismerkmale

… sind selbstverständlich vorausgesetzte Systemmerkmale
… werden vom Benutzer implizit erwartet und oft erst bemerkt, wenn sie fehlen
… muss das System in jedem Fall vollständig erfüllen
… erzeugen keine positive Stimmung, sondern vermeiden lediglich, dass starke Unzufriedenheit entsteht

Leistungsmerkmale

… sind die explizit geforderten Systemmerkmale
… erhöhen die Zufriedenheit mit dem Grad der Erfüllung der Erwartungen
… können sowohl negative als auch positive Emotionen auslösen

Begeisterungsmerkmale

… kennt der Benutzer nicht und werden erst während der Benutzung als angenehme und nützliche Überraschungen entdeckt («Wow»-Effekt)
… bieten einen grossen Nutzen und steigern die Zufriedenheit
… erhöhen den Wert und Eindruck des Produkts und bringen einen Wettbewerbsvorteil
… wecken beim Fehlen keine negativen, beim Vorhandensein jedoch positive Emotionen

Da sich der Benutzer an die Produktmerkmale gewöhnt und Mitbewerber beliebte Merkmale bald auch in ihrem eigenen Produkt anbieten, werden Begeisterungsmerkmale im Laufe der Zeit zu Leistungsfaktoren und schliesslich zu Basismerkmalen. Etwas ursprünglich Besonderes wird zum Standard (s. Abbildung 14). Eine Möglichkeit, Benutzer nachhaltig zufriedenzustellen, sind regelmässige Updates, durch die neue Leistungs- und Begeisterungsmerkmale nachgeliefert werden.

Abbildung 14: Mit der intuitiven «pull-to-refresh»-Geste kann der Benutzer den Inhalt einer Ansicht aktualisieren, indem er mit einem Finger von oben nach unten wischt. Während die «pull-to-refresh»-Geste bei ihrer Einführung 2008 als «cooles» Feature betrachtet wurde, gilt sie heute in vielen Apps als Standard (Screenshots aus Twitter-App).

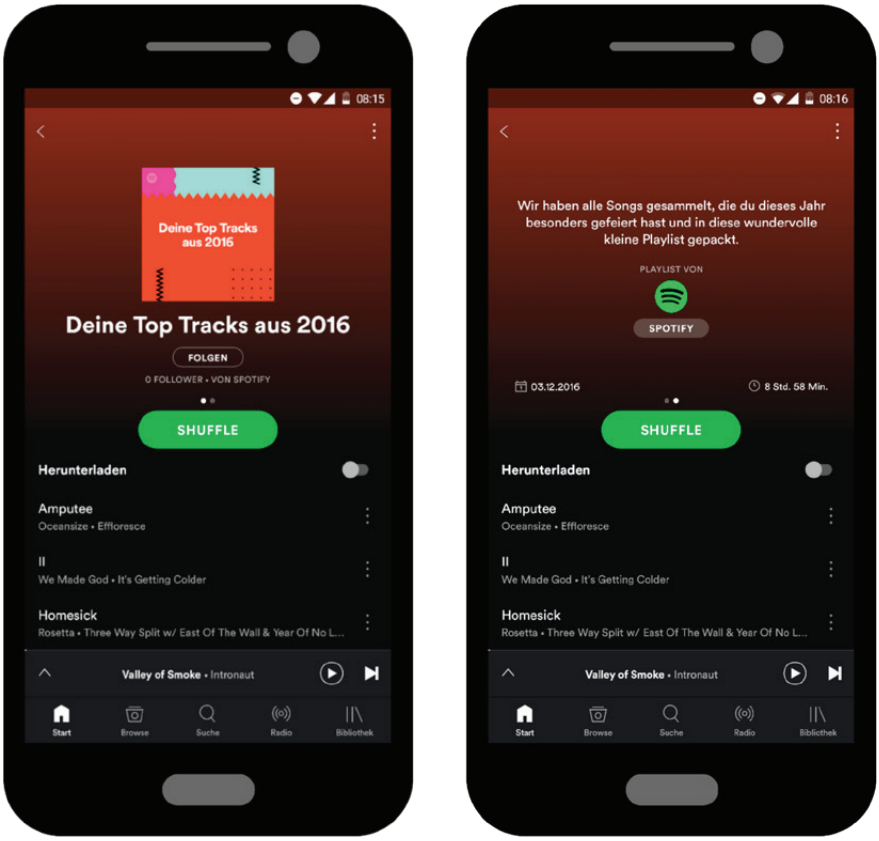


Abbildung 15: Spotify überraschte Ende 2016 mit einer Playlist der persönlichen Top-Tracks des Jahres. Ein unerwartetes und nettes Feature, das positive Emotionen auslöste.

5.4 KONZEPTENTWICKLUNG & PROTOTYPING

Sind die Benutzeranforderungen geklärt, werden zunächst Konzepte entwickelt, wie die Anforderungen an das zu entwickelnde Produkt erfüllt werden können. Anschliessend werden diese Konzepte mithilfe von Prototypen visualisiert und getestet.

Prototypen repräsentieren ein System (oder Teile davon) und bieten Benutzern die Möglichkeit, mit dem System zu interagieren, bevor das finale Produkt existiert. Sie simulieren eine realistische Situation und erleichtern dadurch das Erfassen von Bedürfnissen, Wünschen und Vorstellungen von Benutzern. Zudem bilden Prototypen die Vorstellungen über das zu entwickelnde Produkt explizit ab und schaffen damit eine gemeinsame Diskussionsgrundlage.

Wichtig ist das Erstellen von klickbaren Prototypen, um die Konzepte erlebbar zu machen und Nutzungsprobleme leicht identifizieren zu können. Erst wenn die Screens klickbar sind, wird deutlich, wie sich die App anfühlt, ob sie sich gemäss den Erwartungen verhält und an welchen Stellen das Konzept noch nicht ausgereift genug ist und optimiert werden muss (vgl. Schilling, 2016).

5.5 EVALUATION

Die Evaluation von Prototypen und verschiedenen App-Entwicklungsstadien nimmt in der benutzerzentrierten Entwicklung eine bedeutende Rolle ein. In der Evaluation wird die Benutzersicht zu Bedienung, Verständlichkeit, Struktur und Design eines Produkts erfasst. Schwachstellen werden identifiziert und Ideen zur Optimierung des Produkts gesammelt. Ursachen von Nutzungsproblemen werden ergründet und konkrete, visualisierte Lösungsvorschläge erarbeitet.

Eine bekannte und verbreitete Methode einer Benutzerevaluation ist der Usability Test. In Usability Tests wird die Benutzbarkeit eines Produkts mit Benutzern aus der spezifischen Zielgruppe anhand von vordefinierten Testaufgaben auf Verbesserungspotenzial getestet.

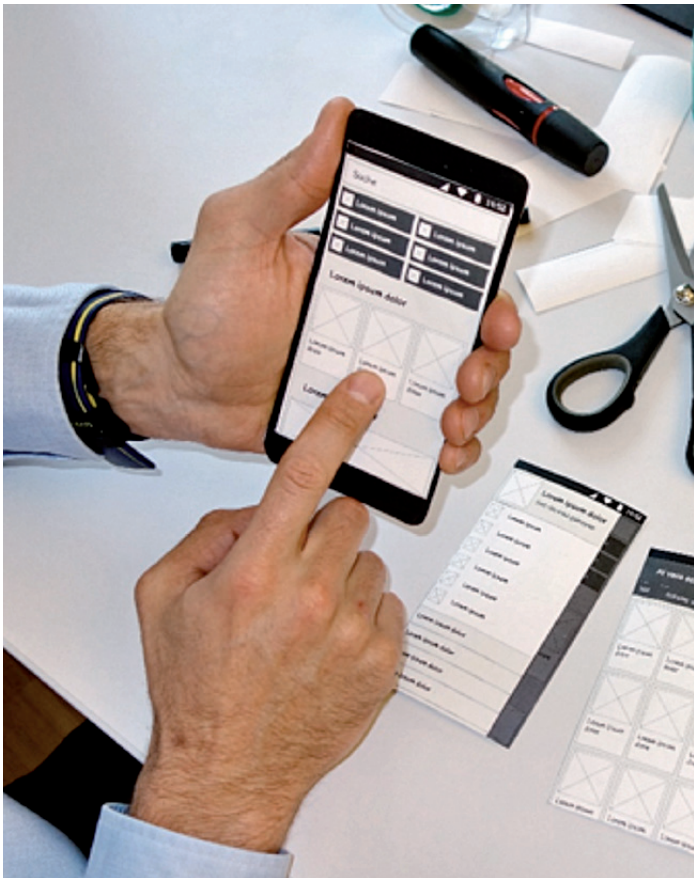


Abbildung 16: Beispiel eines Papierprototypen

5.6 MOBILE-UX-DESIGN-PRINZIPIEN

Um für eine App ein positives Benutzungserlebnis zu gestalten, sollten ein paar Grundregeln beherzigt werden. Das Einhalten der Regeln garantiert noch keinen Erfolg, hilft jedoch, grobe Schnitzer zu vermeiden, welche Benutzer von der Benutzung der App abhalten.

Zweck der App definieren

- Definieren Sie den Zweck der App (Daseinsberechtigung) und formulieren Sie ihren Mehrwert
- Stellen Sie sicher, dass sich der Mehrwert der App sofort erschliesst
- Motivieren Sie die Benutzer bereits beim ersten Erkunden, die App regelmässig benutzen zu wollen

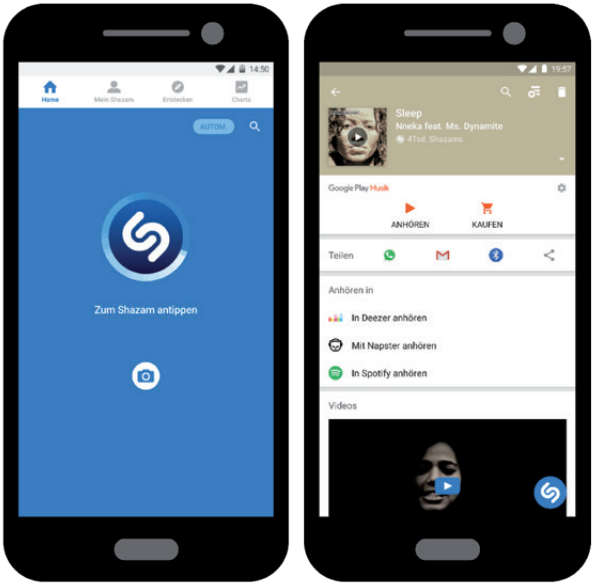


Abbildung 17:
Wer kennt dies nicht: Man hört einen Song, weiss jedoch nicht, wie er heisst und von welchem Interpreten er ist. Genau in dieser Situation liefert die «Shazam»-App einen grossen Nutzen, indem sie eine kurze Audioaufnahme analysiert und sofort die gewünschten Informationen präsentiert.

Zielgruppe und Nutzungsumgebung kennen

- Bestimmen Sie die Merkmale der Benutzer, Aufgaben und der physischen Umgebung
- Identifizieren Sie die Akzeptanzkriterien aus Benutzersicht
- Stellen Sie die Benutzer mit ihren Zielen, Bedürfnissen, Erwartungen und Interessen konsequent in den Mittelpunkt

Funktionsumfang und Bildschirmseiten nicht überfrachten

- Reduzieren Sie den Funktionsumfang und konzentrieren Sie sich auf den Kernnutzen der App
- Verwenden Sie für jede Aufgabe eine eigene Bildschirmseite
- Reduzieren Sie die Anzahl UI-Elemente auf ein Minimum

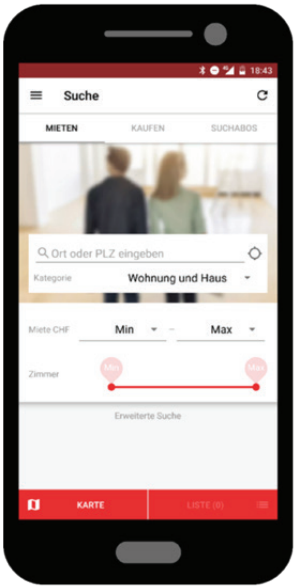


Abbildung 18:
Die Startansicht der «Homegate»-App ist auf das Wesentliche reduziert.

Klare Orientierung und Navigation gewährleisten

- Verteilen Sie die User-Interface-Elemente übersichtlich
- Versuchen Sie ohne Scrolling (insb. horizontales Scrolling) auszukommen
- Gestalten Sie Interaktionselemente gross, gut sichtbar und mit ausreichendem Abstand zu anderen Elementen
- Gestalten Sie Bereiche um kleinere Elemente auch klickbar
- Stellen Sie Inhalte und Funktionen, die für mobile Benutzer besonders interessant sind, priorisiert und prominent dar (z. B. Karte und GPS-Daten, Kontaktdaten, Filialsuche)
- Bieten Sie die Registrierung (falls notwendig) einfach und schnell an

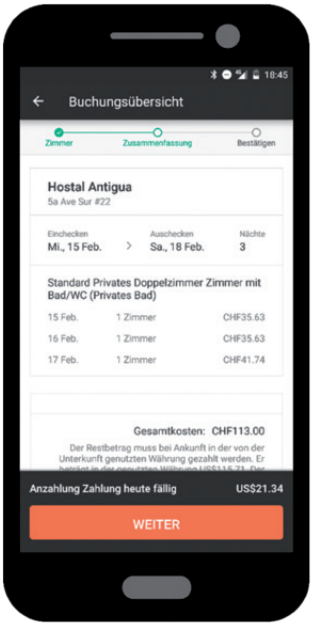


Abbildung 19:
Die «Hostelworld»-App bietet dem Benutzer vor dem Kauf eine verständliche Übersicht und lenkt seinen Blick eindeutig auf den nächsten Schritt.

Eindeutige Benutzerführung und klares Interaktionsdesign erarbeiten

- Führen Sie den Anwender beim Bearbeiten der Elemente in logischer Weise von oben nach unten
- Informieren Sie den Benutzer über den Erfolg oder Misserfolg einer Aktion (z. B. dezente Erfolgsmeldung im oberen Bildschirmbereich, die automatisch wieder verschwindet)
- Unterbrechen Sie Benutzer nur, wenn es absolut notwendig ist und er eine Information unbedingt bestätigen oder eine Entscheidung treffen muss
- Lassen Sie den Benutzer nach einer Unterbrechung dort weitermachen, wo er aufgehört hat
- Bieten Sie die Möglichkeit an, die App an die individuellen Bedürfnisse und Anforderungen des Benutzers anzupassen

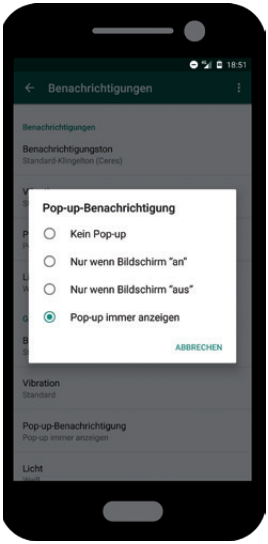


Abbildung 20: Individuelle Einstellungsmöglichkeiten von Benachrichtigungen («Whatsapp»)

Sprache des Benutzers sprechen

- Wählen Sie selbsterklärende Icons und Grafiken
- Verwenden Sie verständliche Begriffe und Texte

Schnelle und flüssige Bedienung garantieren

- Verlegen Sie längere Arbeiten in den Hintergrund
- Komprimieren Sie Bilder, damit sie schnell geladen werden können

Sensorbasierten Funktionalitäten sinnvoll einsetzen

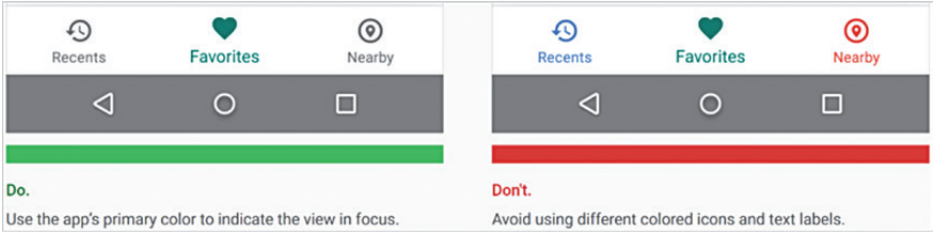
- Setzen Sie Sensoren (Kamera, GPS, Mikrofon, Lagesensor etc.) und Aktoren (Vibration) ein, wenn diese einen Mehrwert bieten
- Integrieren Sie sensorbasierte Daten (z. B. Standort) zu intelligenten Handlungsanweisungen

Abbildung 21:
Sinnvoller Einsatz einer Sensorfunktion: Bei geöffneter SBB-App kann das Smartphone leicht geschüttelt werden, um das letztgekaufte Ticket sofort anzuzeigen. Der Benutzer erreicht sein Ziel effizient, effektiv und zufriedenstellend.

Professionelles Grafik- und Informationsdesign

- Wählen Sie ein passendes und ansprechendes Grafikdesign
- Führen Sie den Blick des Benutzers
- Setzen Sie grosszügige Absätze ein
- Formatieren Sie Text so, dass er «scanbar», also leicht zu überfliegen ist
- Heben Sie wichtige Passagen durch Farbe oder Fettdruck hervor
- Berücksichtigen Sie den jeweiligen Einsatzkontext. Für eine Jogging-App sollten beispielsweise grosse Buttons und mit grossen Abständen verwendet werden, damit die App auch beim Joggen einfach und korrekt bedient werden kann.

Abbildung 22: Beispiel aus Android-Guidelines (Bottom Navigation)[23]



Hersteller-Guidelines berücksichtigen

- Beachten Sie die Design-Prinzipien von Android, iOS und Windows Mobile
- Halten Sie die Do's and Don'ts ein

[23] https://material.io/guidelines/components/bottom-navigation.html#bottom-navigation-style

Benutzer miteinbeziehen

- Ziehen Sie Benutzer aktiv in den Entwicklungsprozess mit ein
- Testen Sie Ihre App mit Benutzern von der Entwurfsphase bis zur Veröffentlichung

Vorteilhafte Präsentation in App Stores

- Verwenden Sie ein attraktives App-Icon
- Formulieren Sie eine klare App-Beschreibung
- Zeigen Sie vielsagende Screenshots

App pflegen

- Gehen Sie auf Kritikpunkte aus dem Benutzerfeedback ein
- Beseitigen Sie Bugs
- Statten Sie die App mit neuen Features aus
- Passen Sie die App an OS-Releases an

6 SOFTWAREENTWICKLUNG

Die Wahl geeigneter Entwicklungswerkzeuge und -praktiken ist eine schwerwiegende Entscheidung zu Beginn der App-Entwicklung. In diesem Kapitel verschaffen wir uns einen Überblick über die verschiedenen Möglichkeiten. Ausserdem betrachten wir zwei der momentan gefragtesten Produkte detaillierter: Xamarin und Ionic.

6.1 SINGLE-PLATFORM

Bei der Single-Platform-Entwicklung wird pro Zielplattform eine eigene Codebasis gepflegt. Dabei werden die Programmiersprachen, Tools und Konzepte der jeweiligen Plattform verwendet (siehe Tabelle 2).

Tabelle 2

	Android	iOS	Windows Mobile
IDE	Android Studio	Xcode	Visual Studio
Programmiersprachen	Java C++	Swift Objective-C	C# C++
UI-Technologie	AXML	Storyboards	XAML

Die Apps, welche beim Single-Platform-Ansatz entstehen, sind sogenannte Native Apps. Ihr grösster Vorteil ist, dass sie aufgrund der Nähe zur Zielplattform sehr performant sind und die vollständige API des darunterliegenden Betriebssystems ausnützen können. Aus demselben Grund sind auch plattformspezifischen Optimierungen der grafischen Oberfläche keine Grenzen gesetzt.

Der grosse Nachteil des Ansatzes: Es müssen mehrere Apps gepflegt werden. Somit muss neue Funktionalität mehrfach in unterschiedlichen Technologien implementiert werden. Oft sieht man in der Praxis, dass sich unterschiedliche Teams um die verschiedenen Plattformen kümmern. In diesem Fall kommen zu den technischen Herausforderungen zusätzliche organisatorische Hürden hinzu.

6.2 CROSS-PLATFORM

Die Idee der Cross-Platform-Entwicklung ist es, mit einer einzigen Codebasis alle Zielplattformen unterstützen zu können. Viele Tools erlauben, neben dem von allen Plattformen genutzten Quellcode, auch plattformspezifische Anpassungen. In diesem Fall hat man zwar nicht mehr 100% «Shared Code», erhält im Gegenzug dafür aber die Möglichkeit für Optimierungen. Beim Cross-Platform-Ansatz entstehen entweder Native, Web oder Hybrid Apps.

Native Apps

Die Apps dieser Kategorie bieten dieselben Vorteile wie die Native Apps der Single-Platform-Entwicklung (Kapitel 6.1), es wird aber zusätzlich deren grösster Nachteil entschärft: Anstelle mehrerer Quellcodes existiert nur noch eine Codebasis für alle Apps.

Web Apps

Web Apps sind keine Apps im eigentlichen Sinne – vielmehr sind es stark optimierte Webseiten, welche den Eindruck von einer App erzeugen sollen. Die Ausführung der App erfolgt im Webbrowser des Betriebssystems.

Die Ausführung im Webbrowser ist Stärke und Schwäche zugleich. So müssen diese Apps nicht wie andere Apps über App Stores verteilt werden und können dadurch sehr einfach aktualisiert werden. Ausserdem kann die App auf jedem Gerät mit verfügbarem Webbrowser gestartet werden – das sollte heutzutage jedes mobile Betriebssystem beherrschen.

Die grössten Nachteile finden sich in der reduzierten Performance, fehlendem plattformspezifischem Look & Feel sowie stark eingeschränktem Zugriff auf Hardwarefunktionen. Der letzte

wurde durch Features von HTML5 zwar ein wenig entschärft, man ist aber weiterhin weit von den Möglichkeiten von Native Apps entfernt.

Hybrid Apps

Apps dieser Kategorie sind ein Mix aus Native und Web-Apps – halt eben ein «Hybrid». Alle Tools, welche Hybrid Apps erzeugen, basieren auf derselben Grundidee: Innerhalb einer Native App befindet sich eine Web View, welche sich um die Ausführung und Darstellung einer Webanwendung kümmert.

Aufgrund des nativen Containers werden Apps dieser Art wie gewohnt über die App-Stores verteilt. Der Container erlaubt ausserdem Zugriff auf die plattformspezifischen APIs. Beides ist mit reinen Web-Apps nicht möglich.

Gegenüber Native Apps weisen Hybrid Apps zwei Nachteile auf: Erstens ist die Performance aufgrund der Ausführung der Webapplikation in der Web View geringer, zweitens muss das plattformspezifische Look & Feel über Styles nachgebildet werden. Dies funktioniert nicht immer wunschgemäss, weshalb Apps dieser Kategorie oft ungewohnt aussehen.

6.3 BEKANNTE CROSS-PLATFORM-PRODUKTE

In Tabelle 3 sind einige bekannte Cross-Platform-Produkte dargestellt. Nachfolgend werden zwei der momentan gefragtesten Produkte – Xamarin und Cordova – näher betrachtet.

Tabelle 3

Produktname	App-Typ(en)	Programmiersprache(n)
Xamarin	Native	C#, XAML
RemObject C#	Native	C#
NativeScript	Native	TypeScript, XML, CSS
Titanium	Native	JavaScript, XML, CSS
Fire Monkey	Native	Delphi, C++
Kony	Native, Hybrid, Web	JavaScript
Unity 3D	Native, Hybrid	C++, Unity Script, C#, Boo
Cordova	Hybrid	JavaScript, HTML, CSS
PhoneGap	Hybrid	JavaScript, HTML, CSS
Ionic	Hybrid	TypeScript, HTML, SASS

Microsoft Xamarin

Xamarin[24] ermöglicht die Entwicklung von Cross-Platform-Native-Apps für Android, iOS und Windows Mobile. Es ist möglich, Anwendungen zu 100% in C# zu erstellen. In der Praxis ist es aber üblich, Teile der grafischen Oberfläche mit den plattformspezifischen Single-Platform-Technologien umzusetzen.

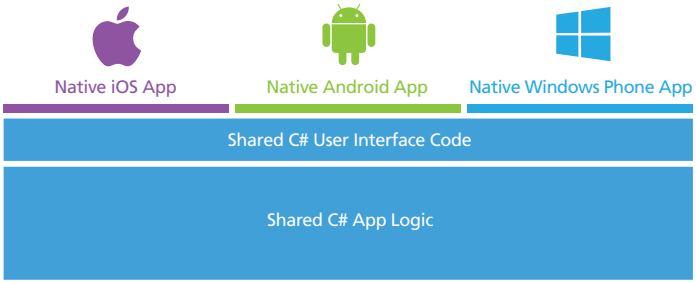


Abbildung 23: Aufbau einer Xamarin-App[25]

Xamarin wurde im Mai 2011 gegründet. Im Februar 2016 kaufte Microsoft die Firma für über 300 Millionen US-Dollar auf. Aufgrund der sehr schlecht laufenden Mobile-Sparte von Microsoft (Kapitel 2.3) vermuten Beobachter, dass Microsoft sich neu vor allem als Anbieter von Entwicklungstools für Mobilplattformen einen Namen machen möchte. Die Integration von Xamarin in Visual Studio ist zumindest jetzt schon weit fortgeschritten.

[23] https://www.xamarin.com/
[25] https://blog.xamarin.com/announcing-xamarin-3/

Abbildung 24:
Logo von Xamarin

Neben den reinen Produkten zur Entwicklung von Apps (Xamarin Platform und Xamarin.Forms) bietet Xamarin auch Produkte für das Testing (Xamarin Test Cloud und Xamarin. UITest) sowie für das Monitoring (Insights bzw. neu HockeyApp) an. Abgerundet wird die Produktpalette durch ein Ausbildungsprogramm mit dem Namen «Xamarin University».

Apache Cordova

Abbildung 25:
Logo von Cordova

Bei Apache Cordova[26] handelt es sich um das bekannteste Framework zur Entwicklung von Cross-Platform-Hybrid-Apps. Cordova stellt ein Open-Source-Framework zur Verfügung, macht aber keinerlei Vorschriften dazu, mit welchen Webtechnologien und -tools die Applikationslogik implementiert werden soll. Um solche Aspekte kümmern sich diverse Distributionen, wie zum Beispiel Ionic[27], die auf der Grundlage von Cordova basieren.

Abbildung 26:
Logo von Ionic

Im Gegensatz zu Apache Cordova enthält das Ionic-Framework Vorgaben dazu, wie die Anwendungslogik strukturiert werden soll. Dazu verwendet es das Angular-Webframework[28] mit JavaScript oder TypeScript als Programmiersprache. Zusätzlich enthält Ionic bereits visuelle Styles und Icons für plattformspezifische Formatierungen der grafischen Oberfläche.

[26] https://cordova.apache.org/
[27] https://ionicframework.com/
[28] https://angular.io/

6.4 PRINZIPIEN UND PRAKTIKEN

In diesem Punkt unterscheidet sich die Entwicklung von Mobile Apps nur wenig von anderen Bereichen. Die gewohnten Prinzipien (Design Patterns, Clean Code, SOLID …) und Praktiken (Unit Testing, Refactoring, CI/CD …) sind ohne Einschränkungen möglich. Einige unterstützende Tools werden in den Kapiteln 7 (Testing) und 9 (Application Lifecycle Management) näher erläutert werden.

Oft wird uns die Frage gestellt, ob die vorgängig genannten Praktiken und Prinzipien für «einfache» Mobile Apps wirklich nötig wären. Es ist tatsächlich so, dass Mobile Apps meist weniger Logik als Desktop-Anwendungen enthalten. Im Gegenzug müssen sie allerdings häufig auf Hardware zugreifen oder Daten aus einem Backend beziehen, wodurch die Komplexität in etwa gleich wie bei anderen Anwendungen ist.

Es lohnt sich somit auch bei Mobile Apps, frühzeitig in ein wartbares Software-Design und qualitätssichernde Massnahmen zu investieren. Besonders ein hoher Grad an Automatisierung ist aufgrund der hohen Änderungsrate im Mobile-Umfeld erstrebenswert. So können wiederholt anfallende Tätigkeiten, z. B. das Ausführen funktionaler Tests beim Erscheinen einer neuen Betriebssystemversion, einfach und komfortabel ausgeführt werden.

7 TESTING

Wie andere Software wollen auch Mobile Apps systematisch getestet werden. Aufgrund der vielfältigen Herausforderungen empfiehlt sich die Erstellung eines Testkonzepts, das Antworten auf folgende Fragen liefert:

- Welche Arten von Tests werden erstellt?
- Welche Tests werden automatisiert?
- Wann werden die Tests ausgeführt?
- Wie werden Testfälle verwaltet?
- Auf welchen Geräten und Betriebssystemversionen wird getestet?
- Auf welchen Testumgebungen wird getestet?

7.1 TESTARTEN

Die meisten aus anderen Softwareprojekten bekannten Testarten können auch für das Testing von Mobile Apps angewendet werden. Um einige wichtige Arten zu nennen:

- Unit-Tests
- Akzeptanz-Tests
- System-Tests
- UI-Tests
- Performance-Tests
- Last Tests

Daneben existieren weitere, spezielle Testarten, die eine zusätzliche Erklärung benötigen:

Usability Test

In Kapitel 5 wurde auf die Wichtigkeit einer angemessenen UX eingegangen. Usability Testing dient der Überprüfung der entworfenen UX-Konzepte.

Feld-Test

Mobile Apps sollten, wann immer möglich, unter realen Bedingungen geprüft werden. Der Nutzungskontext (z. B. Helligkeit, Erschütterungen, Bewegung, Netzqualität und vieles mehr) hat grossen Einfluss auf die Ergebnisse des Tests. Bei Labortests fehlen diese Aspekte z. T. vollständig.

Interrupt Test

Tests dieser Kategorie prüfen, wie sich die App bei einer unvorhergesehenen Unterbrechung verhält. Beispiele für solche Unterbrechungen sind eingehende Telefonanrufe oder Textnachrichten.

Background Test

Der Anwender kann Apps durch Wechsel auf eine andere App in den Hintergrund stellen. Background Tests prüfen, ob eine App korrekt in den Hintergrund versetzt werden kann. Ferner sollte überprüft werden, ob eine App im Hintergrund sparsam mit den verfügbaren Ressourcen umgeht. Ein möglicher Indikator hierfür ist die Messung des Akkuverbrauchs.

Konformitäts-Test

Apps, die in den offiziellen App-Stores vertrieben werden, müssen bestimmte Anforderungen erfüllen. Um unliebsame Überraschungen beim Publizieren von Apps zu vermeiden, ist es sinnvoll, die Einhaltung dieser Kriterien frühzeitig zu prüfen.

Installations- und Update-Test

Die allermeisten Apps erfahren im Laufe ihrer Lebenszeit eine Vielzahl von Updates. Tests dieser Kategorie prüfen die korrekte Aktualisierung einer App bzw. der darin enthaltenen Daten.

7.2 TESTUMGEBUNGEN

Aus Zeit- und schlussendlich Kostengründen ist es unmöglich, die entwickelten Apps auf allen verfügbaren Geräten und Betriebssystemen zu testen. Die Wahl angemessener Testgeräte ist deshalb entscheidend, doch was heisst «angemessen»?

Bei der Beantwortung dieser Frage hilft das Internet, beispielsweise die Website «Perfecto Mobile»[29]. Die Macher dieser Website ermitteln in regelmässigen Abständen, welches die am meisten verwen-

[29] http://tools.perfectomobile.com/test-coverage-optimizer/

deten Geräte und Betriebssystemversionen pro Region sind. Dadurch kann gezielt auf den wichtigsten Geräten des anvisierten Marktes getestet werden und so mit geringem Aufwand eine hohe Abdeckung erreicht werden.

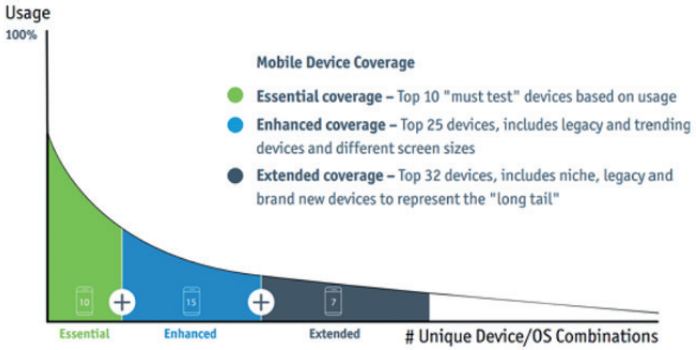


Abbildung 27: Kategorisierung der «Top-Geräte» nach Perfecto Mobile

Nachdem geklärt ist, auf welchen Geräten und Betriebssystemversionen getestet werden soll, steht nun die nächste Frage an: Müssen alle diese Geräte selber gekauft werden? Glücklicherweise nicht zwingend, denn es stehen verschiedene, alternative Testumgebungen bereit:

Gerätepark

Eine naheliegende Lösung ist die Anschaffung eines eigenen Geräteparks. Das macht zu einem gewissen Grad durchaus Sinn, da die Entwickler für die Tests während der Entwicklung ohnehin reale Geräte brauchen. Mit steigender Zahl der Geräte kann diese Variante aber schnell ins Geld gehen, zumal gerade der Unterhalt der Geräte (Aufspielen von Updates etc.) nicht zu unterschätzen ist.

Gerätelabor

Anstelle der Anschaffung eines eigenen Geräteparks lohnt sich unter Umständen das Einmieten bei einem Gerätelabor. Gegen Entgelt können Geräte in öffentlichen Labors gemietet und für Tests verwendet werden. Ein weltweiter Anbieter ist beispielsweise Open Device Lab[30].

Device Cloud

Anstelle eines physischen Zugangs wird bei dieser Variante nur via Webbrowser mit den Geräten in der Cloud interagiert. Bekannte Anbieter wie Xamarin[31] oder Perfecto Mobile[32] bieten gegen Bezahlung Zugang zu Tausenden unterschiedlichen Geräten. Je nach Anbieter kommen entweder reale oder emulierte Geräte zum Einsatz. Bezahlt wird in der Regel nach Nutzungsdauer: Je länger die Geräte verwendet werden, desto mehr kostet die Dienstleistung.

Abbildung 28:
Einige Geräte der Xamarin Test Cloud[33]

[30] https://opendevicelab.com/
[31] https://www.xamarin.com/test-cloud
[32] https://www.perfectomobile.com/learn/mobile-app-testing
[33] https://blog.xamarin.com/build-2015-sessions-tests-and-vs-2015/

Crowd Testing

Beim Crowd Testing wird die App durch eine Masse von Testern über das Internet getestet. Jeder Tester lädt die App auf sein eigenes, privates Gerät und führt bestimmte vorgeschriebene Tests durch. Je nach Anzahl der Tester wird dabei ein entsprechendes Entgelt fällig. Weltweit existieren verschiedenste Anbieter, beispielsweise Testbirds[34].

Emulatoren und Simulatoren

Gerade während der Entwicklung sind reale Geräte oft noch gar nicht nötig. Einfache funktionale Tests können auch auf Emulatoren oder Simulatoren durchgeführt werden. Alle wichtigen Plattformen bieten entsprechende Tools in ihren SDKs an.

Webbrowser

Da Hybrid und Web Apps auf Webtechnologien basieren, können einfache funktionale Tests direkt in einem Webbrowser durchgeführt werden. Dies ist vor allem in einer frühen Phase der Entwicklung interessant, beispielsweise um den Entwicklern rasches Feedback zu liefern.

Bei der Wahl der geeigneten Umgebung sind zwei Aspekte von grundlegender Wichtigkeit:

- Anzahl der benötigten Geräte: Je mehr unterschiedliche Geräte benötigt werden, desto eher sollte eine «öffentliche» Lösung anvisiert werden.
- Sensitivität der App: Je schützenswerter die zu testende App, desto eher sollte eine «private» Lösung anvisiert werden.

[34] https://www.testbirds.de/

Eine Warnung zum Schluss: Nicht alle Testumgebungen eignen sich für alle Testarten. So sind beispielsweise Performance Tests auf Simulatoren, Emulatoren oder Webbrowsern nicht sinnvoll, da diese Systeme sich zu stark von der realen Hardware unterscheiden. Und gewisse Testarten sind erst gar nicht möglich. So fehlt beispielsweise bei einigen Device Clouds die für Interrupt Tests benötigte Funktion, um eingehende Telefonanrufe zu simulieren.

7.3 TESTAUTOMATISIERUNG

Das Testkonzept sollte darüber Auskunft geben, welche der umzusetzenden Testarten automatisiert werden. Aufgrund der hohen Änderungsrate im Mobile-Umfeld sollte ein hoher Automatisierungsgrad angestrebt werden. Dies erlaubt, beispielsweise beim Release eines neuen Geräts oder einer neuen Betriebssystemversion, eine zeit- und kostengünstige Ausführung aller vorhandenen Testfälle.

Aufgrund der Vielfalt an Geräte- und Betriebssystemversionen nehmen automatisierbare UI-Tests bei Mobile Apps einen hohen Stellenwert ein. Bekannte Tools in diesem Bereich sind Xamarin.UITest[35], Appium[36] und Ranorex[37]. Einmal automatisiert kann die Ausführung der Tests auf beliebige Geräte ausgeweitet werden. Die meisten Tools erlauben die Ausführung der UI-Tests auf unterschiedlichen Testumgebungen (Emulatoren und Simulatoren, realen Geräten oder Device Cloud).

[35] https://developer.xamarin.com/guides/testcloud/uitest/
[36] http://appium.io/
[37] http://www.ranorex.com/mobile-automation-testing.html

8 PUBLISHING

Damit die Mobile-Apps von den Benutzern verwendet werden können, müssen diese publiziert werden. Es gibt drei grundsätzlich verschiedene Möglichkeiten zur Verteilung der Apps. Diese werden nachfolgend erläutert.

8.1 APP STORES

Die wohl bekannteste und weitestverbreitete Möglichkeit, um Apps zu veröffentlichen, läuft über App Stores. Für die unterschiedlichen Betriebssysteme gibt es verschiedene Anbieter. Die bekanntesten sind der Apple App Store unter iOS[38] und der Google Play Store unter Android[39].

Für Android-Geräte existieren weitere Stores, wie zum Beispiel der Amazon App Shop[40], Aptiode[41] oder Uptodown[42]. Auch diese Stores enthalten eine Vielzahl von Apps, die auf Android-Geräten installiert werden können. Für iOS gibt es alternative Stores nur mittels «Jailbreak» – also für Geräte, die softwaretechnisch verändert wurden, um diverse einschränkende Restriktionen von Apple zu umgehen. Die bekannteste Alternative ist Cydia[43]. Dieser Store enthält allerdings nur Apps, die von Apple nicht geprüft wurden. Apps aus diesem Store sind deshalb mit grosser Vorsicht zu verwenden.

Für die Veröffentlichung der Apps in den offiziellen Stores müssen zum Teil strenge Anforderungen erfüllt werden. Dadurch ist die Publizierung einer App – gerade für den Apple Store – alles andere als trivial. Zudem ändern sich die Abläufe und Anforderungen oft.

Da die offiziellen App Stores alle hochgeladenen Apps überprüfen, kann es schon mal zu längeren Wartezeiten vor der Veröffentli-

[38] https://itunes.apple.com/ch/genre/ios/
[39] https://play.google.com/store
[40] https://www.amazon.de/appshop_android_app
[41] https://www.aptoide.com/
[42] http://de.uptodown.com/windows
[43] https://cydia.saurik.com/

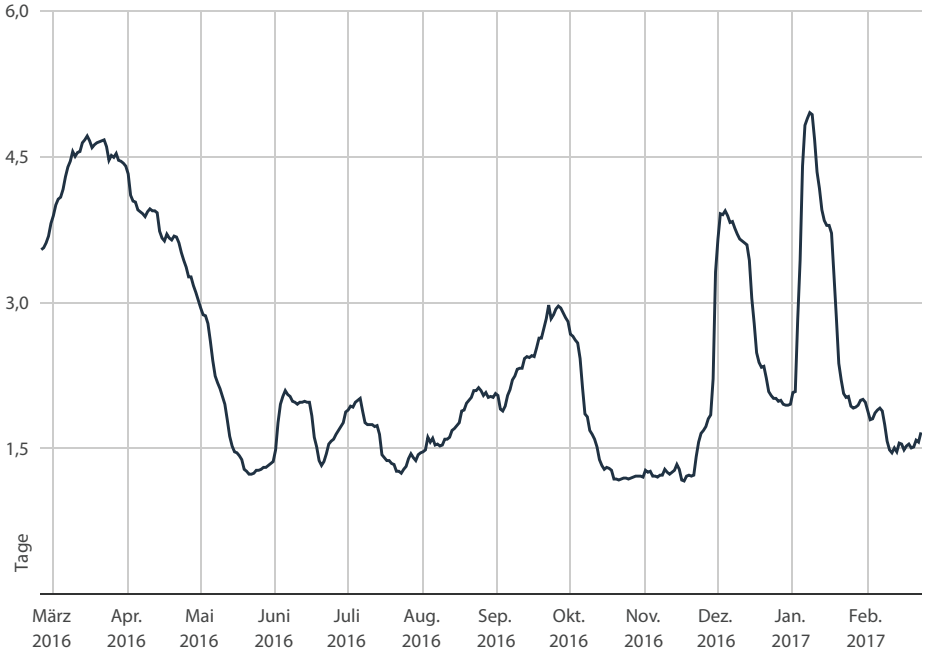


Abbildung 29: iOS App Store – Review-Zeitdauer[44]

chung kommen. Aus diesem Grund sollten alle zu veröffentlichenden Apps vor dem Upload auf Konformität geprüft werden, da es andernfalls zu einer Ablehnung der Anwendung kommen kann. Da Apple die Apps – im Gegensatz zu den anderen Anbietern – manuell überprüft, ist vor allem hier mit längeren Wartezeiten zu rechnen (bis zu zwei Wochen). Erfahrungsgemäss sind die Wartezeiten gegen Ende des Jahres am höchsten. In Abbildung 30 sind die durchschnittlichen Review-Zeiten für iOS-Apps der letzten 12 Monate ersichtlich.

[44] http://appreviewtimes.com/ios/annual-trend-graph

8.2 IN-HOUSE

Viele Apps sollen nur den Mitarbeitern eines Unternehmens zur Verfügung stehen. Da oft sensitive Daten enthalten sind, ist es nicht wünschenswert, dass alle Personen diese App aus den öffentlichen Stores installieren können. In solchen Fällen eignet sich die «In-House»-Verteilung.

Unternehmen können mit unterschiedlichen Softwarelösungen ihre eigenen App Stores verwalten, über welche die Firmen-Apps heruntergeladen werden. Dies bietet weiter den Vorteil, dass die Apps keinen Review-Prozess durchlaufen und somit viel schneller an den Benutzer gebracht werden können.

8.3 AD HOC

Als dritte Option bietet sich die Möglichkeit, Apps «ad hoc» zu verteilen. Man versteht darunter das Verteilen der App per E-Mail oder Website. Dies wird vor allem während der Entwicklung zu Testzwecken angewandt.

Unter Android ist diese Variante sehr einfach realisierbar: Die App kann über einen beliebigen Kanal verteilt und von den Endbenutzern von dort installiert werden. Apple hingegen hat strenge Richtlinien für Apps, die auf den Geräten installiert werden können. Die «ad hoc»-Distribution ist zwar realisierbar, allerdings mit bedeutend mehr Aufwand verbunden als unter Android.

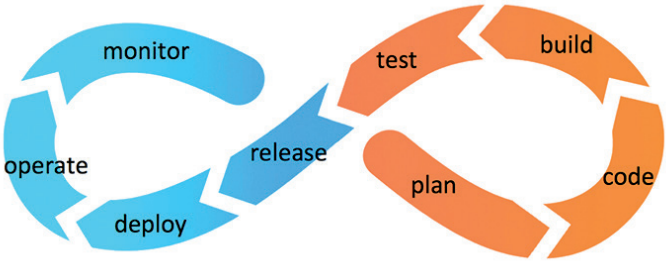
9 APPLICATION LIFECYCLE MANAGEMENT

Unter dem Begriff «Application Lifecycle Management» – kurz ALM – werden verschiedene Massnahmen zusammengefasst, die den Betrieb und Unterhalt der Mobile Apps vereinfachen.

9.1 CI / CD

Wie aus anderen Softwareprojekten gewohnt, existiert auch im Mobile-Umfeld die Möglichkeit zum Aufbau einer Continous-Integration- bzw. Continous-Deployment-Umgebung. Oft benutzte Produkte hierfür sind Jenkins[45] sowie TFS[46] bzw. VSTS[47] von Microsoft.

Abbildung 30:
CI Workflow[48]



CI/CD-Umgebungen bieten verschiedene Vorteile:

- Zeitgewinn durch Automatisierung manueller Tätigkeiten
- Ausschliessen menschlicher Fehler
- Reproduzierbarkeit der Ergebnisse
- Ausführung der Jobs auf einem einheitlichen, neutralen System

[45] https://jenkins.io/
[46] https://www.visualstudio.com/de/tfs/
[47] https://www.visualstudio.com/de/team-services/
[48] http://blog.infostretch.com/why-should-you-use-continuous-integration/

Im Bereich der Mobile Apps empfiehlt sich im Minimum die Automatisierung folgender Tätigkeiten:

- Kompilation der Apps
- Signierung und Publishing der Apps
- Ausführung von Tests
- Statische und dynamische Codeanalyse

Die Automatisierung der oben genannten Schritte entschärft eine der grössten Herausforderungen bei der Entwicklung mobiler Anwendungen: die hohe Änderungsrate. So kann beispielsweise beim Release einer neuen Betriebssystemversion mit minimalem Aufwand ein kompletter Testlauf der App ausgeführt werden.

9.2 ANALYTICS UND MONITORING

Mit der automatisierten Veröffentlichung der Apps sind unsere Möglichkeiten aber noch lange nicht ausgeschöpft. Ein Thema, das zunehmend an Wichtigkeit gewinnt, ist die Analyse des Nutzungsverhaltens der Anwender. Plattformen wie HockeyApp[49] ergänzen Apps mit wenig Aufwand um verschiedene nützliche Funktionen:

- Erhebung von Geräte- und Betriebssysteminformationen
- Erhebung des Nutzungsverhaltens
- Crash-Reporting
- Feedback-Mechanismen

[49] https://hockeyapp.net/

Jegliche Daten werden vom Mobilgerät an ein Backend übermittelt. Websites ermöglichen anschliessend die Auswertung aller Daten. Im Zusammenhang mit dieser «Datenerhebung» müssen einige wichtige Punkte bedacht werden:

- Viele Nutzer sind der Erfassung von Nutzungsdaten gegenüber sehr negativ eingestellt.
- Aufgrund von Datenschutz dürfen nicht beliebige Daten erfasst werden.
- Das Übermitteln von Daten kann den Benutzer Geld kosten, z. B. beim Roaming im Ausland.

Abbildung 31:
Logo von HockeyApp[49]

10 TRENDS

Wie bereits an früherer Stelle erwähnt – Mobile ist nicht nur Smartphone. In diesem abschliessenden Kapitel werden Trends betrachtet, welche in engem Zusammenhang mit dem Thema «Mobile» stehen.

10.1 SEAMLESS INTEGRATION

Heute werden immer mehr Tätigkeiten auf dem Smartphone anstelle des Desktops erledigt. Dennoch ist der Computer für viele Tätigkeiten nicht wegzudenken. So werden zum Beispiel auch heute noch kaum Produkte auf dem Smartphone gekauft. Man stöbert lediglich die Produkte auf dem Smartphone durch, liest dann zum Beispiel die Testberichte auf dem Tablet und kauft das Produkt schlussendlich am Computer.

Der Begriff «Seamless Integration» umfasst die Bemühungen der Hersteller, den Wechsel zwischen den Geräten möglichst komfortabel zu gestalten. Das Ziel ist eine durchgängige und einheitliche Lösung für alle Geräte und Plattformen. Dadurch soll dem Benutzer ein möglichst einfaches und dynamisches Erlebnis geboten werden. Es soll keine Rolle mehr spielen, wo dieser seine Arbeiten erledigen will. Lediglich das, was er tun will, ist noch von Bedeutung.

Bereits heute gibt es dazu verschiedene Produkte wie zum Beispiel Apple Continuity[50] oder Microsoft Continuum[51]. Damit ist es beispielsweise möglich, auf dem Smartphone eine Applikation zu öffnen, seine Arbeiten dort zu beginnen und später auf dem Computer ohne jeglichen Informationsverlust fortzusetzen. Auch vom Computer einen Hotspot auf dem Smartphone zu erstellen, ist somit kein Problem mehr.

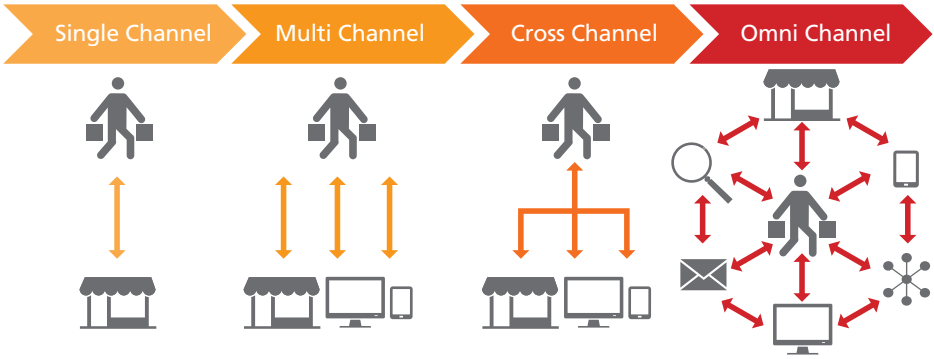
[50] http://www.apple.com/macos/continuity/
[51] https://www.microsoft.com/de-ch/windows/continuums

10.2 OMNI-CHANNEL-MARKETING

Der Online-Handel hat in den letzten Jahren stark zugenommen. Wo man früher in ein Geschäft ging, um ein Produkt zu kaufen, bestellt man dieses heute oft bequem von zu Hause. Die Händler nutzen dies aus, indem sie mehrere Verkaufskanäle anbieten. Wenn die verschiedenen Verkaufskanäle klar voneinander getrennt sind, spricht man von Multi-Channel-Marketing.

Demgegenüber steht Cross Channel Marketing. Dabei gibt es wiederum verschiedene Verkaufskanäle, welche nun aber miteinander verknüpft sind. So ist es zum Beispiel möglich, etwas per Smartphone zu bestellen und dann später im Geschäft abzuholen. Die Weiterentwicklung von Cross-Channel-Marketing ist Omni-Channel-Marketing. Dabei wird der Kunde während des gesamten Kaufprozesses geführt. Ganz unabhängig davon, wann und wo er die einzelnen Schritte wie Informationssuche oder den Verkaufsabschluss durchführen will.

Abbildung 32: Single-, Multi-, Cross- und Omni-Channel-Marketing



Ein spannendes Beispiel aus dem Bereich Omni Channel bietet Disney mit der «MyDisney Experience»[52] an. Diese Dienstleistung unterstützt die Gäste des Disney World vom Planen des Besuchs bis zur Abreise über verschiedene Plattformen hinweg. In einem ersten Schritt ist es möglich, den gesamten Besuch im Park via Website zu planen. Danach kann auf dem Smartphone eine App installiert werden, auf welcher unter anderem die persönlichen Daten abgefragt werden können. Im Disneyland selbst erhält man dann ein Armband – das sog. Magic Band[53] –, mit welchem es unter anderem möglich ist, die Tür des Hotelzimmers aufzuschliessen oder schneller zu den Bahnen zu gelangen. Disney bietet so ein ganzheitliches Erlebnis für einen unvergesslichen Besuch im Disneyland.

10.3 INTERNET OF THINGS UND CLOUD

Internet of Things befasst sich damit, dass die Gegenstände um uns herum immer intelligenter werden. Heute benötigen die verschiedenen Geräte oft unsere Aufmerksamkeit, um sie bedienen zu können. «Intelligente Dinge» sollen das ändern: Sie funktionieren ohne direkte menschliche Interaktion auch im Hintergrund. Erreicht wird dies mit verschiedensten Sensoren und Ausgabegeräten.

In diesem Zusammenhang spielen auch Wearables eine grosse Rolle. Vor allem Smartwatches wurden in den letzten Jahren immer präsenter. Zur jetzigen Zeit dienen diese meist als Erweiterung von Smartphones. Die Entwicklung geht jedoch in

[52] https://disneyworld.disney.go.com/plan/
[53] https://disneyworld.disney.go.com/plan/my-disney-experience/bands-cards/

die Richtung, dass sie immer mehr als eigenständige Geräte dienen. Ein weiteres Anliegen von Benutzern ist es zudem, Körperaktivitäten aufzuzeichnen. Neben den Smartwatches gibt es zu diesem Zweck auch Fitnessbänder oder sogar Polo-Shirts (z. B. «The PoloTech Shirt von Ralph Lauren»[55]), welche alle denkbaren Aktivitäten aufzeichnen.

Nicht nur das Aufzeichnen von Daten ist ein grosses Thema, sondern auch die Sammlung und Interpretation. Um die von IoT-Geräten erfassten Werte möglichst komfortabel zu verwenden, werden diese meist in einer Cloud abgelegt. So können diese gewaltigen Datenmengen komfortabel analysiert und aufbereitet werden. Zu den Möglichkeiten der Cloud empfehlen wir an dieser Stelle das bbv-Booklet «Cloud und Microsoft Azure in a Nutshell».

Abbildung 34:
PoloTech Shirts, welche alle denkbaren Aktivitäten aufzeichnen.[54]

[54] https://cdn2.tnwcdn.com/wp-content/blogs.dir/1/files/2014/08/Ralph-Lauren-Polo-Tech.jpg
[55] http://press.ralphlauren.com/polotech/

10.4 AUGMENTED UND VIRTUAL REALITY

Weiteres grosses Entwicklungspotential liegt in der Augmented bzw. Virtual Reality. Bei Augmented Reality wird die eigene Umgebung mit virtuellen Informationen versehen. Beispiele hierfür sind Google Glass[56] oder Microsoft HoloLens[57]. Damit können zum Beispiel Informationen zu Gegenständen, Gebäuden, aber auch Personen eingeblendet werden. Die Öffentlichkeit reagierte gerade beim letzten Punkt sehr empfindlich: Viele Personen befürchteten, dass sie von «Google Glass»-Trägern beobachtet und überwacht würden. Aufgrund anhaltender Kritik hat Google das Projekt deshalb vorläufig gestoppt, plant aber intern bereits an der Wiederaufnahme der Idee.

Abbildung 35:
Google Glass[58]

[56] https://www.google.com/glass/start/
[57] https://www.microsoft.com/microsoft-hololens/en-us
[58] https://plus.google.com/communities/105306970516704274456/stream/8a6eb099-5582-4dee-bfda-1e25fb225e32

Im Gegensatz zur Augmented Reality erzeugen Virtual-Reality-Systeme eine vollständige virtuelle Welt. Bereits erhältliche Produkte sind die Oculus Rift[59] oder HTC Vive[60]. VR-Produkte werden heute vor allem in Spielen eingesetzt und erzeugen so ein völlig neues Spielerlebnis – sofern einem dabei nicht übel wird. Da das Auge unterschiedliche Bewegungen als der Körper wahrnimmt, leiden viele Anwender unter Übelkeitssymptomen wie bei der Seekrankheit. Aus diesem Grund ist es üblich geworden, zu den Spielen ein «Motion Sickness Rating»[61] mitzuliefern – also die Wahrscheinlichkeit, dass es dem Verwender eines Spiels dabei übel werden könnte.

[59] https://www3.oculus.com/en-us/rift/
[60] https://www.vive.com/de/
[61] http://www.ign.com/wikis/playstation-4/PlayStation_VR_Games_Listed_by_Motion_Sickness_Potential

11 ANHANG

11.1 AUTOREN

Thomas Kälin, M. Sc. FHO in Engineering, entwickelt seit über 15 Jahren Softwaresysteme in den Bereichen Desktop, Web und Mobile. Als Leiter der Mobile Community bei der bbv Software Services AG beschäftigt er sich intensiv mit der Entwicklung von Mobilapplikationen für alle gängigen Plattformen. Zusammen mit Loana Albisser führt er ausserdem die Mobile User Group Zentralschweiz.

Loana Albisser ist seit 2016 Mobile Software-Ingenieurin bei der bbv Software Services AG. Seit ihrem Abschluss an der Hochschule Luzern – Technik und Architektur beschäftigt sie sich mit der Mobileentwicklung und ist stets auf der Suche nach guten neuen Cross-Platform-Ansätzen. Zusammen mit Thomas Kälin führt sie ausserdem die Mobile User Group Zentralschweiz.

Tobias Bregy hat Arbeits- und Organisationspsychologie studiert und bringt bei bbv als User Experience Consultant die Benutzersicht in Entwicklungsprojekte mit ein. Durch Projekte in Branchen wie Telekommunikation, Finanz- und Versicherungsdienstleistungen sowie Behörden und Medizintechnik erwarb erfundiertes Know-how in der Durchführung von Anforderungsanalysen, der Gestaltung von Interaktionskonzepten, der Erstellung von interaktiven Prototypen und der benutzerzentrierten Evaluation.

11.2 QUELLENVERZEICHNIS

Barton, T., Müller, C. & Seel, C. (2016). Mobile Anwendungen in Unternehmen – Konzepte und betriebliche Einsatzszenarien. Wiesbaden: Springer Vieweg.

Burmester, M., Laib, M. & Katharina, S. (2014). User Experience – Positives Erleben betrieblicher IT. Interaktion als positives Erlebnis – Technologiegestaltung neu denken. Abgerufen am 02. 01. 2017 von http://www.mittelstand-digital.de/MD/Redaktion/DE/PDF/wissenschaft-trifft-praxis-ausgabe3,property=pdf,bereich=md,sprache=de,rwb=true.pdf

DIN EN ISO 9241-210. (2010). Ergonomie der Mensch-System-Interaktion – Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme. Brüssel: Europäisches Komitee für Normung.

Fahrenkrug, J. (2014). Mobile User-Experience: App-Richtlinien für bessere Nutzererfahrung. Abgerufen am 26. 12. 2016 von http://t3n.de/magazin/app-richtlinien-bessere-nutzererfahrung-mobile-233331/

Moser, C. (2012). User Experience Design – Mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern. Heidelberg: Springer.

Preim, B. & Raimund, D. (2015). Interaktive Systeme Band 2 – User Interface Engineering, 3D-Interaktion, Natural User Interfaces. Heidelberg: Springer.

Richter, M. & Flückiger, M. D. (2016). Usability und UX kompakt – Produkte für Menschen. Heidelberg: Springer.

Sarodnick, F. & Brau, H. (2011). Methoden der Usability Evaluation. Bern: Hans Huber.

Schilling, K. (2016). Apps machen: Der Kompaktkurs für Designer: Von der Idee bis zum klickbaren Prototyp. München: Hanser.

Schickler, M., Reichert, M., Pryss, R., Schobel, J., Schlee, W. & Langguth, B. (2015). Entwicklung mobiler Apps – Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health. Heidelberg: Springer.

Wächter, M. (2015). Mobile Strategy – Marken- und Unternehmensführung im Angesicht des Mobile Tsunami. Wiesbaden: Springer Gabler.

bbv Software Services AG ist ein Schweizer Software- und Beratungsunternehmen, das Kunden bei der Realisierung ihrer Visionen und Projekte unterstützt. Wir entwickeln individuelle Softwarelösungen und begleiten Kunden mit fundierter Beratung, erstklassigem Software Engineering und langjähriger Branchenerfahrung auf dem Weg zur erfolgreichen Lösung.

Unsere Booklets und vieles mehr finden Sie unter
www.bbv.ch/publikationen

MAKING VISIONS WORK.

www.bbv.ch · info@bbv.ch