Accessible Android Development – Mobile Apps barrierefrei entwickeln

Henrik Voigt, Stefanie Jäckel, Eberhard Zehendner

Institut für Informatik, Friedrich-Schiller-Universität Jena

henrik.voigt@freenet.de, stefanie.jaeckel@informatikideen.de, nez@uni-jena.de

Zusammenfassung

Die Nutzung mobiler Applikationen ergänzt und ersetzt zunehmend das Surfen im Internet. Können Prinzipien der Barrierefreiheit von Webseiten sowie Methoden zu ihrer Herstellung auf die Entwicklung mobiler Applikationen übertragen werden? Und wenn ja, welchen Aufwand erfordert dies? Ausgehend von den *Web Content Accessibility Guidelines 2* der *W3C* setzen die *Google Developer Guidelines* einen Rahmen für die Entwicklung barrierefreier *Android*-Applikationen. Die in Android verfügbaren Systemdienste *TalkBack, BrailleBack, Voice Access* und *Switch Access* unterstützen die konkrete Realisierung. Die vorliegende Arbeit zeigt exemplarisch, was beim Entwurf barrierefreier Android-Apps zu beachten ist, wie die Umsetzung erfolgen kann, welchen Aufwand dies nach sich zieht und wie die Barrierefreiheit einer Implementierung überprüft werden kann. Es lässt sich feststellen, dass barrierefreies Applikationsdesign für Android heutzutage problemlos möglich ist und tatsächlich immer stärker durchgeführt wird.

1 Einleitung

Aufgrund der Nähe der Applikationsentwicklung zur Web-Entwicklung werden für die Konzeption von Projekten häufig Standards aus dem Web-Bereich herangezogen und auf ihnen aufbauend anschließend plattformspezifisch Richtlinien, sogenannte *Developer Guidelines* entwickelt. Richtungsweisend für die barrierefreie Applikationsentwicklung sind die *Web Content Accessibility Guidelines (WCAG)* Version 2.0 (W3C, 2008), seit dem 5. Juni 2018 ergänzt durch die *WCAG 2.1* (W3C, 2018), wobei letztere den Schwerpunkt besonders auf Zugänglichkeit für Menschen mit Seh- oder Lernbeeinträchtigung sowie die Nutzung mobiler Applikationen durch Menschen mit Beeinträchtigungen legen.

Inhaltlich stimmen die *WCAG 2.0* weitgehend mit den geltenden Vorschriften der *Barrierefreie-Informationstechnik-Verordnung (BITV 2.0)* überein. Allerdings wird Barrierefreiheit in Bezug auf die Nutzung mobiler Applikationen vom deutschen Gesetzgeber nur für staatliche

Veröffentlicht durch die Gesellschaft für Informatik e. V. 2018 in
R. Dachselt, G. Weber (Hrsg.):
Mensch und Computer 2018 – Workshopband, 02.–05. September 2018, Dresden.
 https://doi.org/10.18420/muc2018-ws01-0435

Angebote explizit garantiert, und auch dort nur mit gewissen Einschränkungen. Zwar wird die BITV als rechtliche Basis für alle technischen Entwicklungen im Gebiet der Informationstechnik angesehen, auch für mobile Applikationen (Thesmann, 2016, S. 52). Verpflichtend ist die BITV allerdings nur gegenüber Angeboten von Behörden und nicht, wie sicher wünschenswert, gegenüber Unternehmen und Privatpersonen. Und auch die kürzlich verabschiedete Änderung des Behindertengleichstellungsgesetzes (BGG) hat diesen Mangel nicht behoben. In ihrem universellen Anspruch und ihrem unmittelbaren Bezug auch auf mobile Applikationen (W3C, 2015b) gehen die WCAG 2 damit deutlich weiter.

Die WCAG 2 definieren vier von der technischen Umsetzung unabhängige Designprinzipien: Wahrnehmbarkeit, Bedienbarkeit, Verständlichkeit und Robustheit. Jedes der vier Prinzipien wird durch zwölf Richtlinien genauer spezifiziert und jede Richtlinie kann mithilfe von Erfolgskriterien auf erfolgreiche Umsetzung überprüft werden.

Eine Besonderheit der WCAG 2.1 (gegenüber der WCAG 2.0) ist die dort enthaltene Richtlinie 2.5, welche sich den Eingabemodalitäten für Geräte widmet. Barrierefreie Eingabe bedeutet demzufolge nicht ausschließlich die Nutzungsmöglichkeit durch Bedienen einer Tastatur, sondern auch die Einbeziehung moderner Mensch-Maschine-Schnittstellen wie Gestensteuerung oder Spracheingabe. Dies schafft neue Nutzungsszenarien und vergrößert den Kreis potentieller Anwenderinnen und Anwender, erhöht aber gleichzeitig auch die Anforderungen an die Fähigkeiten der Software-Entwicklerinnen und Entwickler.

Ein noch umfassenderer Standard – ob er nun WCAG 3.0 heißen wird oder eine allgemeinere Bezeichnung sich anbietet (Cullipher, 2018) – ist bereits in der Diskussion und wird eventuell auch weitere Technologien wie das *Internet of Things (IoT)* oder digitales Publizieren einbeziehen (Christopherson, 2017).

Am 14. Juni 2018 hat der Deutsche Bundestag, zur Umsetzung der Richtlinie (EU) 2016/2102 über den barrierefreien Zugang zu den Websites und mobilen Anwendungen öffentlicher Stellen in nationales Recht, in zweiter und dritter Beratung die Änderung des Behindertengleichstellungsgesetzes beschlossen (Deutscher Bundestag, 2018, Tagesordnungspunkt 10). Zentral für das im vorliegenden Beitrag behandelte Thema ist die künftige Vorschrift in § 12a Absatz 1 Satz 1 BGG (BT-Drs. 19/2728, 2018): „Öffentliche Stellen des Bundes gestalten ihre Websites und mobilen Anwendungen, einschließlich der für die Beschäftigten bestimmten Angebote im Intranet, barrierefrei.“ Etwas schwächer, aber tendenziell auf das gleiche Ziel hin orientiert, lautet der künftige § 12a Absatz 7 BGG (BT-Drs. 19/2072, 2018): „Der Bund wirkt darauf hin, dass gewerbsmäßige Anbieter von Websites sowie von grafischen Programmoberflächen und mobilen Anwendungen, die mit Mitteln der Informationstechnik dargestellt werden, aufgrund von Zielvereinbarungen nach § 5 Absatz 2 ihre Produkte so gestalten, dass sie barrierefrei genutzt werden können.“

Aufgrund ihrer allgemein gehaltenen Struktur können die Web Content Accessibility Guidelines als eine Art Grundsatzpapier angesehen werden, von dem ausgehend konkrete technologiespezifische Guidelines mit entsprechenden Umsetzungsbeispielen generiert werden können. Für die Entwicklung von Android-Applikationen bieten die *Google Developer Guidelines* (Google, 2018e) eine derartige Funktionalität an.

2 Barrierefreie Applikationen für Android entwickeln

Vor der Entwicklung einer barrierefreien Applikation müssen wir uns mit den vom zugrundeliegenden Betriebssystem zur Verfügung gestellten Diensten der Barrierefreiheit vertraut machen. Im Folgenden beschränken wir uns auf das von Google entwickelte und weiterhin gepflegte Betriebssystem *Android.*

Betriebssystemdienste sind Softwaremodule, die das Betriebssystem standardmäßig seinen Nutzerinnen und Nutzern bereitstellt, ohne dass sie installiert werden müssen. Android bietet für Nutzerinnen und Nutzer mit Einschränkungen die Dienste *TalkBack, BrailleBack, Voice Access* und *Switch Access* an (Google, 2018g). Der Screenreader TalkBack (Google, 2018d) liefert eine Sprachausgabe von Bildschirminhalten. BrailleBack (Google, 2018c) ermöglicht das Anschließen einer Braillezeile an ein Android-fähiges Gerät. Voice Access (Google, 2018h) erlaubt es, mit dem Gerät über einen gewissen Sprachbefehlssatz zu kommunizieren. Switch Access (Google, 2018a) erleichtert Menschen mit physischen Einschränkungen die Bedienung von Benutzeroberflächen, indem es Schnittstellen z. B. für externe Joysticks oder Buttons bedient und über entsprechende Bildschirminhalte Feedback liefert.

Eine barrierefreie Android-App muss nach dringender Empfehlung von Google (2018f) die vier von Android zur Verfügung gestellten Dienste für Menschen mit körperlichen Einschränkungen unterstützen. Zudem empfiehlt Google, die Applikation explizit auf die Funktionalität gegenüber den Diensten TalkBack und Switch Access zu überprüfen, um umfänglichen Zugang für alle zu gewährleisten (Lemcke, 2017).

3 Konzeption barrierefreier Applikationen

Barrierefreie Applikationen zu entwerfen und umzusetzen ist heutzutage zweifelsohne erfolgreich möglich. Es darf dennoch nicht verschwiegen werden, dass sauberes, barrierefreies Design und dessen Umsetzung gegenüber nicht barrierefreier App-Entwicklung immer noch einen Mehraufwand darstellt. Die großen Anbieter, wie Google oder Apple, setzen hier gezielt Entwicklerinnen und Entwickler mit Handicap ein, um Erfahrungen aus erster Hand in den Prozess einfließen zu lassen.

Letzteres transzendiert die für eine klassische partizipative Software-Entwicklung charakteristische Zusammenarbeit zwischen Anwendungs- und Technikexpertinnen und -experten, die maßgeblich geprägt wird durch wechselseitiges Aushandeln und Lernen sowie kreatives Co-Design (Maaß et al., 2016, S. 10): Die Technikexpertinnen und -experten *sind* gleichzeitig Anwendungsexpertinnen und -experten, Aushandlungsprozesse können (zumindest teilweise) entfallen; doch fehlen dann möglicherweise auch die positiven Effekte des Sich-aneinander-Reibens von Menschen, von Erfahrungen, von Auffassungen, von Zielsetzungen.

Auch hier werden die Vorteile einer klassischen partizipativen Software-Entwicklung angestrebt, bei welcher Vertreterinnen und Vertreter aus der zukünftigen Benutzergruppe eng in

den Prozess der Software-Erstellung eingebunden sind: Die erstellten Systeme sollen hinsichtlich ihrer Software-Ergonomie und Benutzbarkeit deutlich besser an die Zielgruppe angepasst sein und ansonsten marginalisierte Perspektiven einbeziehen (Maaß et al., 2016, S. 10).

Aber auch Unternehmen, die keine partizipative Software-Entwicklung praktizieren, was aus wirtschaftlichen Gründen insbesondere auf kleinere Entwicklungsteams zutreffen könnte, sind heutzutage durchaus in der Lage, solide barrierefreie Apps entwerfen; die Community bietet hierfür genügend Hilfestellung an, auch wenn dies zusätzlichen Aufwand bedeutet.

Laut Thesmann (2016, S. 44) sollte zunächst eine uneingeschränkt zugängliche Version entworfen werden. Diese kann dann, dem Ansatz des *Progressive Enhancement* (W3C, 2015a) folgend, schrittweise und je nach gesonderten Anforderungen um weitere Technologien erweitert werden. Die zusätzlichen Technologien müssen dabei nicht unbedingt barrierefrei sein, da sie nur als Alternative zu den bereits implementierten barrierefreien grundsätzlichen Mechanismen dienen. Insgesamt bleibt nämlich „der Webauftritt auch ohne diese Technologien grundsätzlich nutzbar, wenn auch vielleicht stellenweise nicht ganz so komfortabel oder in der Präsentation weniger attraktiv“.

Designerinnen und Designern müssen die Zugangskanäle zum jeweiligen Angebot klar sein. Sie müssen wissen, wie sie Sehbehinderten ihre Angebote geeignet zur Verfügung stellen, z. B. als Audioalternative oder mit funktionierender TalkBack-Anbindung. Außerdem gilt es, für Hörgeschädigte das Angebot auf ausreichenden Zugang in Form von Schrift oder Gebärden zu erweitern. Menschen mit physischen Einschränkungen muss eine Schnittstelle zur leichten Navigation ermöglicht werden (etwa über eine Anbindung an Switch Access). Als allgemeine Regel gilt, die Inhalte so detailliert abzuklopfen, dass geeignete alternative Zugänge bestehen und für niemand eine Informationsbarriere auftritt. Kurzum – die App bietet ihren gesamten Funktions- und Informationsumfang für alle Zugreifenden an.

Dieses optimistische Ziel ist leider nicht für alle Entwicklungen zu erreichen, da manchmal aufgrund der starken Spezialisierung der App, wie etwa bei der App *Instagram* für das Teilen von Fotos, nur unzureichend Alternativen geschaffen werden können. Nichtsdestotrotz sollten bei der Konzeption zunächst alle Kanäle in Betracht gezogen und nach Möglichkeit in die Umsetzung eingeplant werden.

In Verbindung damit ist es nützlich, sich frühzeitig Gedanken zu machen, wie eine Anbindung an die vier genannten Systemdienste, die Android bereitstellt, umgesetzt werden kann. Dabei lohnt es sich, die gut geführten *Developer Guidelines* (Google, 2018b) zu nutzen, bei der Implementierung von Funktionalitäten auf *Best Practices* zu setzen und bei Fragen die Community im Forum zu Rate zu ziehen.

4 Praktische Umsetzung anhand eines Beispiels

Als Beispiel für eine Implementierung soll eine Planer-App dienen, die eine *Button Navigation* anbietet, *Todos* anzeigt, eine Einkaufsliste speichert und Informationen über das aktuelle Wetter gibt (GitHub, 2018). Dabei wird darauf geachtet, für wesentliche Elemente eine *Content*

Description bereitzustellen, eine *Focus Order* festzulegen und inhaltlich zusammengehörige Elemente in Groupings zusammenzufassen. Auf die Implementierung der Funktionalität hinter der Oberfläche wird hier verzichtet, da der Fokus auf dem Design liegt.

Das vorliegende Beispiel steht als personalisierte Variante einer von Google bereitgestellten Vorlage frei zur Verfügung. Das komplette Listing der XML-Layout-Datei der Beispiel-App ist unter https://users.fmi.uni-jena.de/~nez/Konferenzen/MuC2018/activity_main.xml abrufbar und kann Entwickelnden als Einstieg in die barrierefreie Applikationserstellung dienen.

Zunächst wird ein einfaches relatives Layout erstellt, auf dem alle benötigten Elemente, also die *Buttons*, die *Checkboxen*, das Bild und die Liste platziert werden. Eine Veranschaulichung der intendierten grafischen Oberfläche liefert Abbildung 1. Anschließend muss überlegt werden, welche Elemente in der Navigation eine wesentliche Rolle spielen und ob diese eine *Content Description* benötigen. Bei den Buttons ist dies beim *Löschen-Button* der Fall, da dieser ein Bild verwendet, das für visuell eingeschränkte Menschen nicht erkennbar ist. In die Content Description wird also *Löschen-Button* eingetragen. Selbiges gilt auch für den *Info-Button*.

Abbildung 1: Benutzeroberfläche der Beispiel-App

Die Checkboxen, die anschließend folgen, benötigen keine Description, da sie sowieso durch nebenstehende Labels ausreichend gekennzeichnet sind. Sollten Checkboxen allerdings nicht gelabelt sein, so müssen unbedingt Zugehörigkeit und Beschreibung angegeben werden. Als nächstes wird das Wetterbild dargestellt. Der Image Viewer, welcher das Bild enthält, benötigt eine Content Description, da auf dem Bild Informationen dargestellt werden, die Sehbehinderten sonst verwehrt blieben. Das zugehörige Temperaturbanner braucht hingegen nicht extra gelabelt werden, da im Kontext erkennbar ist, dass es zum Bild gehört und dieses genauer beschreibt. Die Liste am Ende der App benötigt ebenfalls keine Description, da die eingebauten Text Views alle selbsterklärend sind.

Im nächsten Schritt wird die Navigation auf der Oberfläche durch einen gelenkten Fokus festgelegt. Dazu werden in jedes Element, das fokussierbar sein soll, Tags *nextFocusDown* und *nextFocusUp* eingefügt, in denen beschrieben wird, welches Element auf das aktuell angesteuerte Element nach unten bzw. oben folgt (das entsprechende Tag entfällt, wenn es keinen Nachfolger bzw. Vorgänger gibt). Nachdem dies für alle Elemente festgelegt wurde, ist nun ein Fokusfaden entstanden, der von TalkBack und von Switch Access aufgegriffen werden kann und die Anwenderin oder den Anwender sicher durch das Programm führt. Nachfolgender Ausschnitt aus dem XML-Code der Beispiel-Applikation verdeutlicht diese Vorgehensweise:

```
<Button
        android:id="@+id/composeButton"
        android:nextFocusDown="@+id/discardButton"
        … />

<ImageButton
        android:id="@+id/discardButton"
        android:contentDescription="@string/discardButtonDescription"
        android:nextFocusDown="@+id/infoButton"
        android:nextFocusUp="@+id/composeButton"
        … />

<ImageButton
        android:id="@+id/infoButton"
        android:contentDescription="@string/infoButtonDescription"
        android:nextFocusDown="@+id/jetpackCheckbox"
        android:nextFocusUp="@+id/discardButton"
        … />

<CheckBox
        android:id="@+id/jetpackCheckbox"
        android:nextFocusDown="@+id/hyperspaceCheckbox"
        android:nextFocusUp="@+id/infoButton"
        … />
```

Als letzten Punkt soll noch die Verwendung von Groupings gezeigt werden. Dazu befindet sich am unteren Ende der Oberfläche ein *Relative Layout*, welches als Datencontainer fungiert. Darin befinden sich die einzelnen *Text Views* mit den Listeneinträgen. Würde nun nichts weiter angegeben, so würden TalkBack und Switch Access über alle Elemente einzeln iterieren und sie einzeln vorlesen bzw. in den Fokus rücken. Da aber deutlich ist, dass alle Elemente thematisch zusammengehören, nämlich zur Liste, können sie gruppiert und den Nutzenden so als logische Einheit präsentiert werden. Dies erhöht das Verständnis deutlich und erleichtert die Navigation. Im Datencontainer wird das Tag *focusable* eingefügt und auf *true* gesetzt, siehe nachfolgenden XML-Code. Damit wird den Systemdiensten verdeutlicht, dass alles, was sich im Container befindet, als eine logische Einheit anzusehen ist und gemeinsam interpretiert werden kann. Die Elemente des Containers werden nun von TalkBack nacheinander vorgelesen und wie ein einziges Element in den Fokus gerückt.

```
<RelativeLayout
        android:id="@+id/data_container"
        android:focusable="true"
        android:nextFocusUp="@+id/temperatureText"
        …>

        <TextView
                android:id="@+id/ueschrift"
                android:text="Wunschzettel"
                … />

        <TextView
                android:id="@+id/geschenk1"
                android:text="Winterjacke"
                … />

        <TextView
                android:id="@+id/geschenk2"
                android:text="Kinogutschein"
                … />

        <TextView
                android:id="@+id/geschenk3"
                android:text="Handy"
                … />
</RelativeLayout>
```

5 Applikationen auf Barrierefreiheit prüfen

Was macht eine barrierefreie App aus? Nach der Konzeption und Umsetzung der Prinzipien Wahrnehmbarkeit, Bedienbarkeit, Verständlichkeit und Robustheit in einer Applikation gilt es, die Qualität dieser Implementierung zu bewerten. In erster Linie sichert die solide Umsetzung der vier Grundprinzipien einen Großteil der Qualität einer barrierefreien Anwendung. Allerdings kann auch hier weiter differenziert werden. Die Technische Universität Dortmund bietet für den deutschsprachigen Raum einen ausführlichen Qualitätskriterienkatalog (Fakultät Rehabilitationswissenschaften, 2017) an. Auf diesen verweist z. B. das barrierefreie Portal „Digital informiert - im Job integriert“, kurz Di-Ji (FTB, 2017). Dieses Portal bietet für den deutschsprachigen Raum auch umfangreiche Verweise auf Entwicklerreferenzen, Checklisten, Richtlinien, Gesetze und Verordnungen – ist also für den Entwurf und die Einschätzung von mobilen Applikationen sehr hilfreich.

Der Qualitätskriterienkatalog ist im Netz kostenlos erhältlich (Reh@pp-Quality, 2016). Er wird charakterisiert als „vollständige Sammlung aller App Qualitätskriterien, durch die eine App für eine heterogene Nutzer_innen-Gruppe qualitativ hochwertig wird“ (Fakultät Rehabilitationswissenschaften, 2017). Im Katalog findet sich eine übersichtliche Auflistung aller Kriterien, gegenüber welchen eine App bewertet werden kann. Zugleich werden die entsprechenden Normen aufgeführt, an denen die Kriterien jeweils festgemacht wurden. In einem Review Meeting kann dieser Katalog dann genutzt werden, um das Verhalten der App gegenüber den gelisteten Kriterien einzuschätzen und schlussendlich zu bewerten. Qualität im Kontext von barrierefreien mobilen Applikationen zielt hier besonders auf Kundenzufriedenheit, Benutzbarkeit, Bedarfskonformität und Gebrauchsfertigkeit ab. Deshalb bezieht der Katalog neben den vier oben genannten Grundprinzipien aus den Web Content Accessibility Guidelines auch weitere Kriterien wie Zugänglichkeit, Zuverlässigkeit, Medien, Oberflächen-Ästhetik und Erlernbarkeit in die Bewertung ein.

Jeweils ein Qualitätskriterium wird zunächst definiert und erklärt. Anschließend kann durch Aufgreifen der Unterpunkte eine Applikation aus verschiedenen Blickwinkeln betrachtet und schlussendlich die Qualität der Umsetzung bezüglich der Kriterien bewertet werden. Zusätzlich finden sich Referenzen zu offiziellen Normen oder Richtlinien. Anhand der vorgestellten Applikation könnte beispielhaft eine Betrachtung hinsichtlich der Bedienbarkeit durchgeführt werden. Als Unterpunkte werden Einzelaspekte wie Steuerung, Lesbarkeit bis hin zum Glossar aufgeführt. Beginnend mit einer Einschätzung der Steuerung der Applikation, liefert der Kriterienkatalog ein umfangreiches Muster, wie eine optimale Umsetzung auszusehen hat. So heißt es, dass ein intuitiv aufgebautes Menü vorhanden sein sollte, klare und überschaubare Funktionen angeboten werden müssen und Wege innerhalb der App einfach und ersichtlich sind. Anschließend wird die App anhand des Musters eingeschätzt. Ist der Zustand zufriedenstellend, kann zum nächsten Kriterium übergegangen werden, andernfalls wird Unzureichendes korrigiert. Der Katalog ist ein sehr umfangreiches Werk und eignet sich hervorragend, um Applikationen im Kontext der barrierefreien Benutzbarkeit einzuordnen und zu bewerten. Mit diesen Hilfsmitteln ist es auch Laien möglich, Applikationen hinsichtlich der Barrierefreiheit hinreichend zu begutachten und einzuschätzen. Aber auch für erfahrene Entwicklerinnen und

Entwickler ist es hilfreich zu sehen, welche Qualitätskriterien bei der eigenen Entwicklung in Betracht gezogen werden sollten.

6 Fazit und Ausblick

Es lässt sich feststellen, dass barrierefreies Applikationsdesign für Android heutzutage problemlos möglich ist, aber dennoch einen Mehraufwand in der Entwicklung darstellt. Über Gesetze, Normen und Developer Guidelines wird immer erfolgreicher versucht, die Aufmerksamkeit bei der Entwicklung auch auf Aspekte der mobilen Barrierefreiheit zu legen. Es bleibt zu hoffen, dass nicht zuletzt aufgrund der hohen Zahl an Betroffenen und der daraus resultierenden Nachfrage die Prinzipien barrierefreier App-Entwicklung in den Konzeptionsprozess zukünftiger Apps Einzug erhalten. Erst wenn der Fokus bei der Implementierung auf Zugangsbarrieren bezüglich Navigation und Inhalten gelenkt werden kann, werden regelmäßig Lösungen entstehen, die den Anforderungen der Barrierefreiheit genügen. Es muss ein Bewusstsein kultiviert werden, dass (und warum) barrierefreies App-Design richtig und wichtig ist, um den Mehraufwand, den es zweifelsohne mit sich bringt, zu rechtfertigen.

Als erfreulich kann die zunehmende Entwicklung von unterstützenden Apps für Menschen mit Einschränkungen gesehen werden. Zahlreiche mobile Apps zielen exakt auf den Aspekt der Barrierefreiheit ab und ermöglichen so ihren Nutzerinnen und Nutzern, die Welt ein Stück weit mehr zu erleben, als dies ohne das Tool möglich wäre. Apps wie die Microsoft-Entwicklung Seeing AI (Microsoft, 2018) ermöglichen beispielsweise sehbehinderten Menschen eine großartige Erleichterung ihres Alltags durch das Erkennen und auditive Beschreiben der Umgebung mittels moderner Kamera- und KI-Technologie. Diesen Menschen wird durch die Technologie ein Stück Sehvermögen (zurück-)gegeben, Barrieren im Alltag werden abgebaut. Hier steckt großes Potential, und dieser Bereich lässt noch einiges erwarten.

Zwiespältig bleibt der juristische Aspekt. Es muss geklärt werden, wie mit Speicherung und Verarbeitung von persönlichen Daten Dritter, die teilweise sogar unfreiwillig mit diesen Hilfsmitteln konfrontiert werden, umgegangen werden kann.

In Zukunft wird es spannend zu sehen, wie und inwiefern Mensch und Maschine immer reibungsloser miteinander agieren. Der mobilen App wird dabei die Rolle des Informationskonverters zuteil. Mithilfe unterstützender Technologien, wie künstlicher Intelligenz und Big-Data-Techniken, beispielsweise Data Mining und Machine Learning, kann Menschen mit Einschränkungen ein Stück einer ihnen bisher nicht zugänglichen Welt geöffnet werden. Dies ermöglicht Teilhabe, wie im Falle von Seeing AI.

Literaturverzeichnis

BT-Drs. 19/2072. *Drucksache des Deutschen Bundestages 19/2072 vom 9. Mai 2018.*

BT-Drs. 19/2728. (2018). *Drucksache des Deutschen Bundestages 19/2728 vom 13. Juni 2018.*

Christopherson, R. (2017). *WCAG2.1: Web guidelines set for makeover, enabling better mobile and web access for disabled users.* AbilityNet, Blog vom 24. April 2017. Zugriff am 06.07.2018. Verfügbar unter https://www.abilitynet.org.uk/web-global-guidelines-updated-inclusive-disabled-people

Cullipher, V. (2018). *What Comes after WCAG? Jeanne Spellman on W3C's "Project Silver".* Interview mit Jeanne Spellman, geführt am 13. April 2018 von Vivian Cullipher, Microassist. Zugriff am 06.07.2018. Verfügbar unter https://www.microassist.com/digital-accessibility/what-comes-after-wcag-spellman-project-silver/

Deutscher Bundestag. (2018). *39. Sitzung des Deutschen Bundestages am Donnerstag, dem 14. Juni 2018.* Amtliches Protokoll.

Fakultät Rehabilitationswissenschaften, TU Dortmund. (2017). *App-Qualitätskriterienkatalog.* Zugriff am 06.06.2018. Verfügbar unter http://www.rehatechnologie.fk13.tu-dortmund.de/rehapp/de/Toolkit/App-QKK/index.html

Forschungsinstitut Technologie und Behinderung (FTB) und Rehabilitationstechnologie. (2017). *Entwicklung barrierefreier Apps.* Zugriff am 06.06.2018. Verfügbar unter http://www.di-ji.de/index.php?option=com_content&view=article&id=333&Itemid=159&lang=de

GitHub, Inc. (2018). *googlesamples/android-BasicAccessibility.* Zugriff am 06.06.2018. Verfügbar unter https://github.com/googlesamples/android-BasicAccessibility/

Google LLC. (2018a). *About Switch Access for Android. Android Accessibility Help.* Zugriff am 06.06.2018. Verfügbar unter https://support.google.com/accessibility/android/answer/6122836?hl=en

Google LLC. (2018b). *Accessibility overview. Android Developers.* Zugriff am 06.06.2018. Verfügbar unter https://developer.android.com/guide/topics/ui/accessibility/

Google LLC. (2018c). *Google BrailleBack. Apps bei Google Play.* Zugriff am 06.06.2018. Verfügbar unter https://play.google.com/store/apps/details?id=com.googlecode.eyesfree.brailleback&h%20l=de

Google LLC. (2018d). *Google TalkBack. Apps bei Google Play.* Zugriff am 06.06.2018. Verfügbar unter https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback&hl=de

Google LLC. (2018e). *Make apps more accessible. Android Developers.* Zugriff am 06.06.2018. Verfügbar unter https://developer.android.com/guide/topics/ui/accessibility/apps

Google LLC. (2018f). *Test your app's accessibility. Android Developers.* Zugriff am 06.06.2018. Verfügbar unter https://developer.android.com/training/accessibility/testing

Google LLC. (2018g). *Übersicht über die Android-Bedienungshilfen. Hilfe für Android Accessibility.* Zugriff am 06.06.2018. Verfügbar unter https://support.google.com/accessibility/android/answer/6006564?hl=de&ref_topic=6007234

Google LLC. (2018h). *Voice Access. Apps bei Google Play.* Zugriff am 06.06.2018. Verfügbar unter https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.voiceaccess&hl=de

Lemcke, M. (2017). *Barrierefreie Apps, App-Entwicklung mit Android – Grundlagen* [Webinar]. Zugriff am 22.11.2017. Verfügbar unter https://www.youtube.com/watch?v=x4qvhHCdrYQ

Maaß, S., Schirmer, C., Bötcher, A., Buchmüller, S., Koch, D. & Schumacher, R. (2016). *Partizipative Entwicklung von Technologien für und mit ältere/n Menschen. Abschlussbericht zum Forschungsprojekt ParTec – Partizipatives Vorgehen bei der Entwicklung von Technologien für den demografischen Wandel.* Zugriff am 06.07.2018. Verfügbar unter http://nbn-resolving.de/urn:nbn:de:gbv:46-00105568-18

Microsoft. (2018). *Seeing AI. Talking camera app for those with a visual impairment.* Zugriff am 06.06.2018. Verfügbar unter https://www.microsoft.com/en-us/seeing-ai/

Reh@pp-Quality. (2016). *App-QKK. App-Qualitätskriterienkatalog.* Verfügbar unter http://www.reha-technologie.fk13.tu-dortmund.de/rehapp/Medienpool/Dateien-zum-Download/App-QKK.pdf

Thesmann, S. (2016). *Interface Design. Usability, User Experience und Accessibility im Web gestalten* (2. Aufl.). Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-658-03857-1

World Wide Web Consortium (W3C). (2008). *Web Content Accessibility Guidelines (WCAG) 2.0. W3C Recommendation 11 December 2008.* Zugriff am 06.06.2018. Verfügbar unter https://www.w3.org/TR/WCAG20/

World Wide Web Consortium (W3C). (2015a). *Graceful degradation versus progressive enhancement. W3C Wiki.* Zugriff am 06.06.2018. Verfügbar unter https://www.w3.org/wiki/Graceful_degradation_versus_progressive_enhancement#Graceful_degradation_and_progressive_enhancement_in_a_nutshell

World Wide Web Consortium (W3C). (2015b). *Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile. W3C First Public Working Draft 26 February 2015.* Zugriff am 06.06.2018. Verfügbar unter https://www.w3.org/TR/mobile-accessibility-mapping/

World Wide Web Consortium (W3C). (2018). *Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation 05 June 2018.* Zugriff am 06.07.2018. Verfügbar unter https://www.w3.org/TR/WCAG21/

Danksagung

Die detaillierten Überarbeitungshinweise aus dem Begutachtungsprozess der MuC 2018 waren für uns sehr hilfreich. Sie erlaubten uns, die vorliegende Arbeit auf den neuesten Stand zu bringen, ihre Inhalte zu konkretisieren und ihre Themen zu vertiefen.

Autorinnen und Autoren

Voigt, Henrik

Henrik Voigt wurde 1995 in Erfurt geboren, wuchs nahe Erfurt auf und legte das Abitur 2014 in Gebesee ab. Er studierte ab Oktober 2014 Ingenieurinformatik an der TU Ilmenau und wechselte später in den Bachelorstudiengang Informatik an der Friedrich-Schiller-Universität Jena. Aktuelle Studienschwerpunkte sind Software-Entwicklung und

Agile Methoden. Bezüge zur Applikationsentwicklung bestehen durch Projekte in der Freizeit und am Fraunhofer-Institut für Angewandte Optik und Feinmechanik IOF in Jena.

Jäckel, Stefanie

Stefanie Jäckel (Jg. 1984) ist Gymnasiallehrerin für die Fächer Informatik, Geschichte und den Kurs Medienkunde sowie Fachleiterin für Informatik. Bereits während ihres Studiums setzte sie sich kritisch mit Wechselwirkungen zwischen Informatik und Gesellschaft auseinander. Als wissenschaftliche Mitarbeiterin an der Friedrich-Schiller-Universität Jena beschäftigt sie sich im Rahmen ihrer Lehre mit der Sensibilisierung von Studierenden für Inhalte aus dem Bereich *Informatik und Gesellschaft* – insbesondere auch für *Barrierefreiheit im Internet.*

Zehendner, Eberhard

Eberhard Zehendner (Jg. 1956) ist Professor für Technische Informatik an der Friedrich-Schiller-Universität Jena. Seit zwei Jahrzehnten lehrt er u. a. Inhalte aus dem Bereich *Informatik und Gesellschaft,* seit 2012 auch vor Schülerinnen und Schülern im Rahmen der Kinderuniversität sowie jährlich in einem Kurs der Thüringer JuniorAkademie. Derzeit beschäftigt er sich sehr intensiv mit Themen aus den Bereichen alter(n)sgerechte Informatik, Inklusion und Teilhabe. Er ist Vorstandsmitglied des *Forum InformatikerInnen für Frieden und gesellschaftliche Verantwortung (FIfF)* und Mitherausgeber der *FIfF-Kommunikation* – Zeitschrift für Informatik und Gesellschaft.