Paul Neuhold, BSc

Mobile Applikationen für große Nachschlagewerke

Entwicklung eines Prototypen am Beispiel Austria-Forum.org

Masterarbeit

zur Erlangung des akademischen Grades
Diplom-Ingenieur
Masterstudium Softwareentwicklung - Wirtschaft

eingereicht an der

Technischen Universität Graz

Betreuer:

Priv.-Doz. Dipl.-Ing. Dr.techn. Martin Ebner
Institute of Interactive Systems and Data Science

27. Februar 2017

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum	Unterschrift

Abstract

In this research work an evaluation of selected mobile applications from the corresponding app-stores will be done. For this work primarily applications which are made for encyclopedias and fullfilling some given prerequisites are selected. The aim of it is to find out how information is presented and which ways of information retrieval are offered to users.
Furthermore a new smart way will be introduced in which we can deal with knowledge with a special focus to encyclopedias and mobile devices. The prototyped application will carry out special features, which makes usage of mobile devices. The presented approach follows a slightly more aggressive way on how to bring knowledge to the users. With the given permission of the user this solution access the location of him/ her and based on that information relevant articles which are related to the current position of the user will be suggested. To reach the interest of the user communication between users and device is done with the notification system of the corresponding operating systems. Additionally also possibilities will be introduced in order to alter the behaviour of this feature to fit the current needs of the user. Those affect the frequency of pushed notifications and power usage of the device.
The outcome is a prototype with a set of features which allows users to retrieve information based on their mood and their willingness of spending effort. If he/ she is eager to learn the application provides ways to support that. If he/ she just need some quick information there are ways to get it. And if he/ she is interested in a new way of information retrieval via mobile applications for encyclopedias the prototype offers the above mentioned possibility to do so.

Kurzfassung

Im Rahmen dieser Masterarbeit wird zu Beginn eine Evaluierung von vorhandenen mobilen Applikationen für Nachschlagewerke, welche nach bestimmten Kriterien ausgewählt worden sind, durchgeführt. Gegenstand der Evaluierung ist zum einen die visuelle Darstellung der Inhalte und zum anderen die Art und Weise wie Informationen zur Verfügung gestellt werden. Anhand der dadurch gewonnenen Erkenntnisse wurden einige Features definiert und ausgearbeitet. Diese werden in weiterer Folge im Laufe dieser Arbeit vorgestellt.

Das Hauptaugenmerk liegt dabei auf einer neuartigen Methode, welche im Rahmen dieser Masterarbeit entwickelt worden ist, mit der Wissen mit Fokus auf mobile Endgeräte verbreitet werden kann. Wenn Benutzerinnen und Benutzer dieser Methode zustimmen, dann werden sie auf passivem Wege mit Informationen beliefert. Die so verbreiteten Informationen sind jeweils abhängig von den aktuellen Positionen der Benutzerinnen und Benutzer. Zusätzlich werden auch Möglichkeiten erläutert, mit deren Hilfe Benutzerinnen und Benutzer die Verhaltensweisen der jeweiligen Funktionen an ihre momentanen Bedürfnisse anpassen können.

Als Ergebnis wird ein Prototyp vorgestellt, welcher eine Vielzahl an Features anbietet. Diese sind in erster Linie zur Informationsgewinnung bestimmt und bieten je nach aktuellem Bedarf eine Vielzahl an Einstellungsmöglichkeiten, sodass Benutzerinnen und Benutzer unterschiedliche Möglichkeiten haben ihren derzeitigen Wissensdurst zu stillen, sei es das Verlangen nach etwas Unbekannten, relevante Information zur aktuellen Position oder Zeit, oder eine zielgerichtete Suche.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einleitung

Bekannte große Enzyklopädien haben mittlerweile die Produktion und den Vertrieb ihrer gedruckten Ausgaben eingestellt. Brockhaus zum Beispiel gibt es regulär druckfrisch seit dem Jahr 2014 nicht mehr zu kaufen (vgl. Kramer, 2014).
Ein weiteres großes Nachschlagewerk, Encyclopedia Britannica, hat 2012 die Herstellung ihrer gedruckten Versionen eingestellt. Jorge Cauz, Präsident der Encyclopedia Britannica sagt, dass dieser Schritt nichts mit Wikipedia oder Google zu tun hat, sondern vielmehr damit, dass ihre eigene digitale Version oft verkauft wird.[1]
Nichtsdestotrotz wird freie Information in der Zukunft so wie in den letzten Jahren in Quantität sowie Qualität steigen. Wikipedia, als großer Vertreter der Web-2.0-Technologien, trägt maßgeblich dazu bei. Zusätzlich zur Masse an Informationen ist auch der Zugang letzlich um ein Vielfaches einfacher geworden. In der heutigen Zeit, in der fast jeder ein Smartphone besitzt, ist das Abrufen von Information überall und zu jeder Zeit möglich. Genauer gesagt haben 95% aller Jugendlichen im Alter zwischen 12 und 19 Jahren ein Smartphone (vgl. Feierabend et al., 2015, Seite 7).
Um das Bedürfnis nach freien Informationen zu befriedigen und um einen möglichst schnellen und unkomplizierten Zugang zu gewähren sind zahlreiche Lösungen entwickelt worden. In den jeweiligen Stores gibt es einige mobile Applikationen, die darauf abzielen. Diese unterscheiden sich nicht nur in ihrer optischen Darstellung sondern auch im Umfang der Funktionalitäten und der Art und Weise, wie Information präsentiert und angeboten wird.
Ziel dieser Arbeit ist es Möglichkeiten und Funktionen zu diskutieren, zu erforschen sowie festzustellen, die eine mobile Applikation für große Nachschlagewerke bieten soll um den Wissensdurst der Benutzerinnen und Benutzer zu stillen, vor allem wenn die der Applikation zugrunde liegenden Datenquelle ebenfalls durch eine responsive Webseite aufbereitet und zur Verfügung gestellt ist.
Zu Beginn werden die allgemeine Verbreitung von Smartphones und die technischen Möglichkeiten mobile Applikationen umzusetzen thematisiert. Anschließend werden aus den jeweiligen App-Stores ausgewählte Applikationen vorgestellt und evaluiert. Zusätzlich zur eigentlichen Beschreibung dieser wird begründet, wie und weshalb genau diese ausgewählt wurden um sie in die durchgeführte Evaluierung aufzunehmen.

[1] `http://www.telegraph.co.uk/culture/books/9142412/Encyclopaedia-Britannica-stops-printing-after-more-than-200-years.html` zuletzt gesehen am 12.01.2017 um 21:17

Nach der Darstellung des Standes der Technik wird eine offensivere Art und Weise der Informationsverbreitung vorgestellt. Diese Methode soll Benutzerinnen und Benutzer auf passivem Wege mit Informationen versorgen. Der beschriebene Ansatz benötigt dazu allerdings die Berechtigung des/ der einzelnen Benutzers/ Benutzerin um auf deren Position zugreifen zu dürfen. Anschließend werden sie mit Informationen, welche relevant für ihren aktuellen Standort sind, versorgt. Beobachtungen zeigen, dass Benutzerinnen und Benutzer sehr stark auf Benachrichtigungen reagieren oder zumindest einen Blick auf ihr Smartphone werfen, wenn eine solche eintrifft. Genau aus diesem Grund, wird auf die zuvor angesprochenen ortsspezifischen Informationen mittels Benachrichtigung aufmerksam gemacht.
Zusätzlich werden auch Möglichkeiten beschrieben und angeboten, mit deren Hilfe Benutzerinnen und Benutzer das eben genannte Verhalten personalisieren können, damit die Benachrichtigungen schlussendlich kein Aufdringlichkeitsgefühl auslösen. Benutzerinnen und Benutzer können dadurch ebenfalls entscheiden, ob sie eher häufiger benachrichtigt werden wollen oder sich doch lieber für eine energieeffizientere Einstellung entscheiden. Abgeschlossen wird diese Masterarbeit durch eine Diskussion und mit einer Zusammenfassung.

2 Stand der Technik

In diesem Kapitel wird ein Überblick über mobile Applikationen gegeben, welche in Googles Play-Store and Apples App-Store verfügbar sind. Neben der Erfassung des Angebots werden zuerst die Nutzung von Smartphones und anschließend die verfügbaren Technologien im Sinne einer Entwicklung beschrieben.

2.1 Verbreitung des Smartphones[2]

Smartphones werden heutzutage mehr denn je verwendet. Dies wird unter anderem durch eine Studie von Deloitte unterstrichen, welche die Nutzung durch Personen in den Vereinigten Staaten analysiert hat. Laut dieser Studie überprüfen Menschen aller Altersgruppen ihr Handy bis zu 46 mal am Tag. Die Prüfung richtet sich in erster Linie auf Neuigkeiten, die eventuell eingetroffen sind (vgl. Eadicicco, 2015).
Solch eine exzessive Nutzung tritt nicht nur in den Vereinigten Staaten auf. In Japan beispielsweise verwenden Personen im Alter von 14 bis 25 Jahren ihr Handy durchschnittlich zwei Stunden pro Tag. Diese ausgeprägte Verwendung wird sogar noch von Nutzerinnen und Nutzer in Südkorea übertroffen. In diesem Land ist es üblich, dass Smartphones bis zu viereinhalb Stunden pro Tag verwendet werden (vgl. Neidhart, 2015).
Auch in Europa wird viel Zeit für den Umgang mit mobilen Geräten aufgewendet. An dieser Stelle ist die JIM-Studie zu erwähnen, bei der Jugendliche im Alter zwischen 12 und 19 Jahren zu ihrem Nutzungsverhalten und wie sie selbst mit Medien interagieren, befragt worden sind. 92 Prozent der befragten Personen besitzen demnach ein Smartphone und 97 Prozent verwenden dieses mehrmals in der Woche. Eine große Menge, genauer gesagt 75 Prozent, hat auch Zugang zu einer Internetflatrate und ist dementsprechend auch „always on“ (vgl. Feierabend et al., 2015, Seite 6-7, Seite 12, Seite 46).
Eine weitere Studie, durchgeführt von britischen Psychologen, gibt sogar an, dass junge Erwachsene ihr Handy durchschnittlich fünf Stunden pro Tag verwenden. (vgl. Gregoire, 2015)
Zusätzlich zu der jetzt schon intensiven Nutzung ist festzustellen, dass die Zeit die Personen täglich mit ihrem Smartphone verbringen, durch ein stetiges Wachstum

[2]Tw. eingereicht zur Veröffentlichung in Neuhold et al. (2017)

charakterisiert ist. Ein signifkanter Teil davon entsteht durch die Nutzung und den Umgang mit mobilen Applikationen. Allerdings ist auch anzumerken, dass dieses Wachstum zuletzt etwas abgeflacht ist. 2012 konnte noch ein 90,9 prozentiger Anstieg, von 46 auf durchschnittlich 135 Minuten täglich, verzeichnet werden. Im Jahr 2015 ging dieser auf 11,3% zurück, genauer gesagt konnte ein Anstieg von lediglich 17 Minuten verzeichnet werden (vgl. eMarketer, 2015).
Dass die aufgewendete Zeit irgendwann nicht mehr zunehmen wird, ist eine logische Schlussfolgerung. Denn die Tatsache, dass ein Tag nur 24 Stunden hat, wird früher oder später dazu führen.
Neben der täglichen Nutzung von Smartphones erteilt auch ein großer Teil der Benutzerinnen und Benutzer die Erlaubnis, dass installierte Applikationen mit Hilfe von Push-Benachrichtigungen auf sich aufmerksam machen dürfen. Wird eine Applikation installiert, erteilen 41% der iOS Benutzerinnen und Benutzer die Erlaubnis, dass die eben installierte Applikation Benachrichtigungen versenden darf. Unter Android ist diese Abfrage nicht erforderlich. In weiterer Folge ziehen gepushte Mitteilungen nicht immer eine Reaktion nach sich. Genauer gesagt, interagieren nur 8,7 Prozent der Benutzerinnen und Benutzer mit empfangenen Push-Benachrichtigungen (vgl. Shaul, 2016).
Dies mag auf den ersten Blick wenig erscheinen, doch im Verhältnis zur globalen Verteilung von Smartphones, welche für 2017 mit 2.292,5 Millionen vorhergesagt wurde, sind 8,7 Prozent immer noch eine große Menge. Abbildung 1 zeigt das angesprochene Wachstum. Dementsprechend ist ein Markt für Features, welche mit Benachrichtigungen arbeiten, durchaus verfügbar. Allerdings muss an dieser Stelle auch erwähnt werden, dass der Umgang mit solcher mit Bedacht durchzuführen ist. Der Grund für die Behauptung, dass eine vorsichtige Verwendung und Versendung von Benachrichtigungen zu beachten ist, ist die Tatsache, dass Benutzerinnen und Benutzer bei zu häufigem Empfang von Benachrichtigungen diese als störend und lästig empfinden. Ähnlich sieht es auch O'Connell (2016) wenn sie schreibt - „Over 50% of App Users Find Push Notifications Annoying". Ob es sich bei einer Benachrichtigung um eine lokale oder eine remote, welche in der Regel als „Push-Notification" bezeichnet wird, handelt, ist für Benutzerinnen und Benutzer nicht zu unterscheiden. Lokale Benachrichtigungen werden von der Applikation selbst erstellt und konfiguriert. Eine so erstelle Benachrichtigung wird anschließend an das Betriebssystem übergeben, welches zuständig für das Aussenden jener ist. Bei „Push-Notification" wird der In-

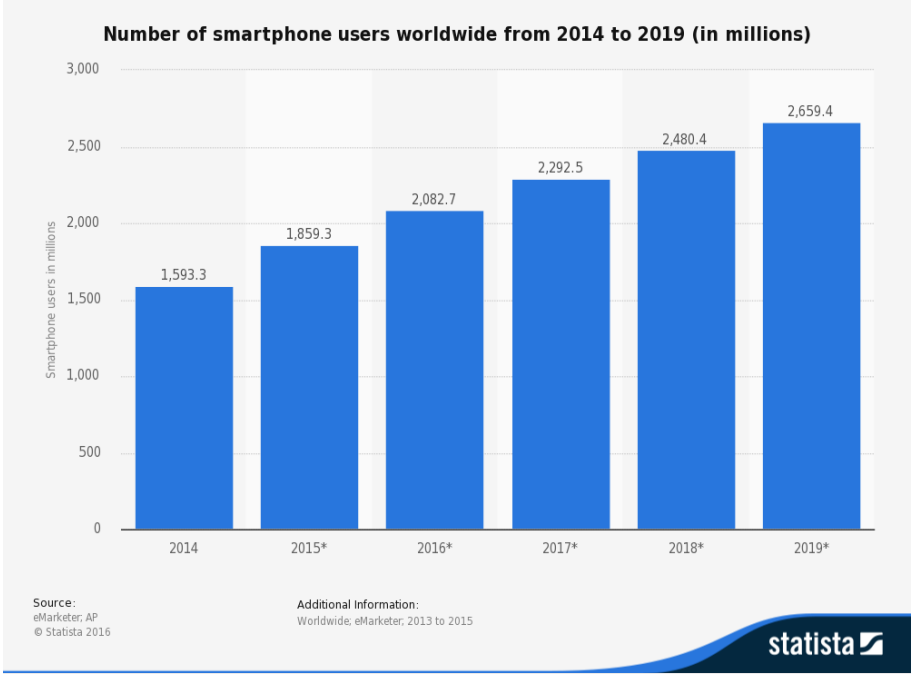

Abbildung 1: Anzahl der Smartphones weltweit (vgl. eMarketer; AP, 2015)

halt einer Benachrichtigung auf einem externen Server erstellt. Auch der Zeitpunkt, wann diese versendet wird, wird vom Server bestimmt. Das eigentliche mobile Endgerät fungiert in diesem Fall als Empfänger für Nachrichten vom Server. Wenn eine Benachrichtigung, egal welcher Art, empfangen wurde, ist die jeweilige Applikation zuständig für die korrekte Bearbeitung der empfangenen Nachrichten[3].

2.2 Technische Möglichkeiten: Native Applikation oder Responsive Webseite für mobile Smartphones

Grundsätzlich gibt es mehrere Lösungsansätze um Benutzerinnen und Benutzer auf ihren Smartphones zu erreichen. Meist wird von einer nativen mobilen Applikation, einer mobilen bzw. responsiven Webseite oder einer hybriden Applikation gesprochen. Eine generelle Aussage zu treffen oder einen pauschalen Vergleich zwischen den genannten Möglichkeiten zu ziehen, ist ohne genaueres Abstecken der Rahmenbedingungen schwer durchzuführen. Denn im Grunde ist die Entscheidung, welche Technologie verwendet werden soll abhängig von den umzusetzenden Zielvorstel-

[3] `https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/` zuletzt aufgerufen am 15.02.2017 22:08

lungen. Um die passende Variante zu wählen, müssen die Anforderungen notiert und anhand der jeweiligen Vor- und Nachteile der einzelnen Varianten abgewogen werden. Demnach bestimmen die Anforderungen sehr stark die zu wählende Technologie. Es gibt natürlich auch Szenarien, bei denen die Wahl eindeutig ist. Um dies etwas genauer zu verstehen werden in folgendem die jeweiligen Möglichkeiten genauer beschrieben.

Mobile Webseite / Responsive Webseite

Von einer mobilen Webseite wird dann gesprochen, wenn es sich um eine Version der Webseite handelt, die ausschließlich für die Verwendung durch mobile Endgeräte entwickelt worden ist. Bei Bedarf können hier gezielt Inhalte für Benutzerinnen und Benutzer angezeigt werden. Bei Webseiten mit responsivem Webdesign handelt es sich um solche, die für jedes Gerät entwickelt und optimiert worden sind. Bei diesen passt sich das Webseitendesign der Größe des jeweiligen Bildschirms an (vgl. Atkin, 2015).

Ob mobile oder responsive Webseite, für beide sprechen beispielsweise Faktoren, wie Erreichbarkeit oder Ausbaufähigkeit bzw. Änderungsmöglichkeit. Wenn es die Dringlichkeit erfordert, sind diese von jedem/ jeder Nutzer/ Nutzerin sofort erreichbar und können mit einem Browser aufgerufen werden. Mobile Applikationen hingegen müssen erst installiert werden. Die Webseite an sich kann jederzeit und ohne entstehenden Aufwand für den/ die Nutzer/ Nutzerin erweitert und ausgebaut werden. Dementsprechend können Fehler sehr schnell und ohne Mitwirken des/ der Benutzers/ Benutzerin behoben werden (vgl. Summerfield, 2016).

Bei nativen Applikationen hingegen erfordert jede noch so kleine Änderung die Erstellung einer neuen Version des Programms. Wird diese anschließend in den dafür zuständigen Store gestellt, müssen Benutzerinnen und Benutzer ihre installierte Applikation erst updaten, bevor die Änderung wirksam wird. Sind automatische Updates deaktiviert, können Änderungen erst sehr spät bis gar nicht beim/ bei der Benutzer/ Benutzerin ankommen. Dadurch gestaltet es sich sehr schwierig jede einzelne anwendende Person mit Updates zu erreichen.

Native mobile Applikation

Native Applikationen machen vor allem dann Sinn, wenn das Ziel beispielsweise ein interaktives Spiel ist oder explizit auf die Hardware sowie Sensoren des Gerätes

zugegriffen wird. Weitere Punkte, die für eine native Applikation sprechen, sind Anwendungsfälle, bei denen es primär um komplexe Berechnungen geht, wenn große Datenmengen verarbeitet werden müssen oder wenn eine regelmäßige Benutzung der Funktionalitäten mit zusätzlicher Personalisierungsmöglichkeit der Applikation im Vordergrund steht (vgl. Summerfield, 2016).
Einer der wichtigsten Punkte, aufgrund derer eine native mobile Applikation gewählt werden sollte, ist die Möglichkeit die angebotene Funktionalität ohne vorhandene Internetverbindung bereitzustellen. Denn jede Webseite benötigt in der Regel eine Verbindung zum Internet um mit dem Gerät zu kommunizieren. Besteht eine Anforderung darin, dass auch die Verwendung der Applikation ohne aktive Internetverbindung möglich sein soll, dann ist es zwingend erforderlich eine native Applikation zu entwickeln.
Neben der Offline-Verfügbarkeit spielt auch die Möglichkeit Benachrichtigungen zu empfangen eine zentrale Rolle. Es gibt zwar Möglichkeiten für mobile Geräte auf Android Basis Benachrichtigungen ausgehend von Webseiten zu empfangen[4]. Für iOS ist diese Funktionalität leider noch nicht verfügbar (vgl. Deglin, 2015).
Dies wird auch in der offiziellen Dokumentation von Apple mit folgendem Hinweis vermerkt: „Notifications for websites do not appear on iOS“[5]. Besteht eine Anforderung darin, dass mit Benachrichtigungen alle Plattformen erreicht werden sollen, dann ist es zwingend notwendig auf eine native Lösung zu setzen.

Hybride Applikation

Neben nativen Applikationen und mobilen bzw. responsiven Webseiten sind an dieser Stelle auch die so genannten hybriden Applikationen zu erwähnen. Hybride Applikationen ermöglichen es den Inhalt von Webseiten in Form einer nativen Applikation wiederzugeben. Webseiten genießen dadurch mehr Privilegien hinsichtlich lokaler Ressourcen oder Zugriff auf Sensoren (vgl. Appel, 2016).
Dies geschieht durch das Rendering der Webseite in einer dafür vorhergesehene View, welche in der Applikation eingebunden ist. Nebst der Darstellung der Webseite beinhaltet eine hybride Applikation auch weitere durch native Entwicklung verfügbar ge-

[4] `https://developers.google.com/web/fundamentals/engage-and-retain/push-notifications/` zuletzt aufgerufen am 02.12.2016 um 9:34

[5] `https://developer.apple.com/library/mac/documentation/NetworkingInternet/Conceptual/NotificationProgrammingGuideForWebsites/Introduction/Introduction.html` zuletzt aufgerufen am 02.12.2016 um 9:35

machte Features. Die Beschreibung einer hybriden Applikation ist in weiterem Verlauf dieser Arbeit auch auf die Begriffe Wikipedia-Reader und Wikipedia-Container zu übertragen.

2.3 Vorhandene mobile Applikationen für Nachschlagewerke

Folgend werden mobile Applikationen für Nachschlagewerke vorgestellt. Ebenfalls wird erläutert nach welchen Kriterien diese ausgewählt worden sind.

2.3.1 Auswahlkriterien

Betriebssysteme
Begründet durch die Verbreitung mobiler Betriebssysteme und der Anzahl an angebotenen mobilen Applikationen, wurde beschlossen, dass im Rahmen dieser Arbeit lediglich solche für Android und iOS betrachtet werden. Denn wie Pryjda (2016) schreibt, tut sich Windows schwer und verliert fast überall. Android dominiert klar den deutschsprachigen Markt hinsichtlich der Verbreitung mobiler Betriebssysteme. Wird allerdings der amerikanische Markt betrachtet so ist zu erkennen, dass hier die Dominanz von Android nicht ganz so ausgeprägt ist. In diesen Ländern liefern sich Apples iOS und Google's Android ein Kopf-an-Kopf-Rennen. (vgl. Pryjda, 2016; Keleva, 2016)
Durch Abbildung 2 wird ersichtlich, dass Android auch global gesehen deutlich vor allen anderen Betriebssystemen liegt. Lediglich iOS spielt noch eine wichtige Rolle bei der Verteilung. Das, wie oben schon erwähnte, schwächelnde Windows und andere Systeme sind kaum verbreitet und spielen deshalb für die durchgeführte Evaluierung keine Rolle.
Abbildung 3 zeigt auf, dass trotz der dominierenden Verbreitung von Android, die Anzahl an verfügbaren Applikationen in den oben genannten Stores sehr ausgeglichen ist. Auf Grund dessen liegt es auf der Hand, dass für die folgende Evaluierung Applikationen für iOS und Android betrachtet werden.

2.3.2 Suchkriterien

Grundsätzlich hat der/ die Benutzer/ Benutzerin viele Möglichkeiten um auf eine Applikation in Apple-App-Store bzw. Google-Play Store aufmerksam zu werden.

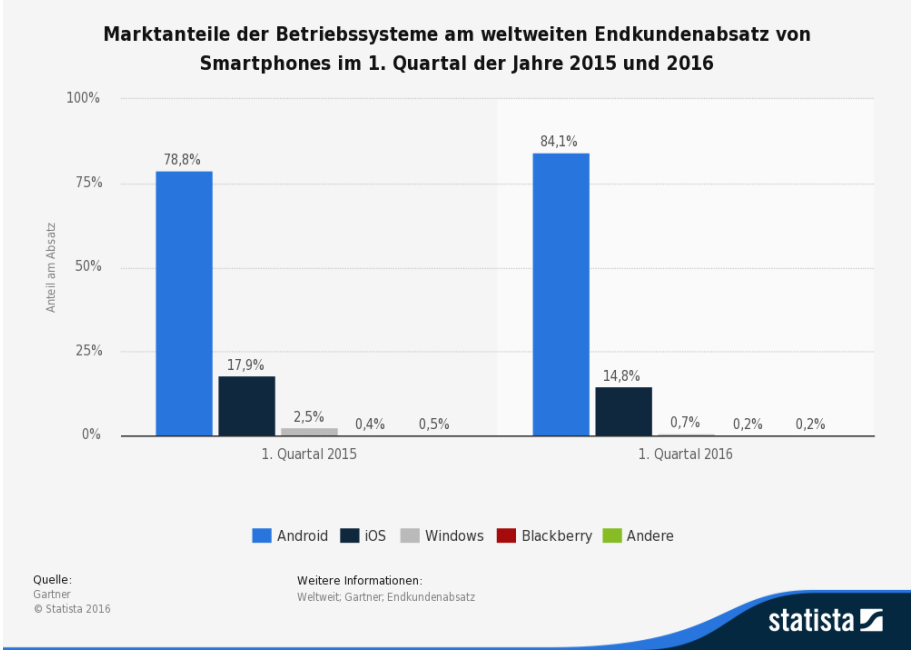

Abbildung 2: Weltweite Marktanteile der Betriebsysteme Q1 2015/2016

Kategorien wie „Diese Woche angesagt“ oder „Top-Titel der Woche“ versuchen die Aufmerksamkeit der Benutzerinnen und Benutzer auf der Startseite zu erhalten. Für die getätigte Auswahl an Applikationen im Rahmen dieser Arbeit spielen diese „eye-catcher“[6] Kategorien allerdings keine Rolle. Für die Auswahl wird von der Annahme ausgegangen, dass Benutzerinnen und Benutzer aktiv eine Applikation für Nachschlagewerke, Enzyklopädien bzw. basierend auf einem Wiki[7] suchen. In Anbetracht dessen wurde die direkte Suche mit folgenden Begriffen gewählt:

- Enzyklopädie, encyclopedia
- Wikipedia
- Wiki
- Brockhaus
- Encyclopedia Britannica

Die dadurch resultierenden Suchergebnisse wurden anhand nachfolgender Auswahlkriterien analysiert und eingestuft. Neben dem Namen der Applikation wurde

[6] `https://de.wikipedia.org/wiki/Blickfang` zuletzt aufgerufen am 30.11.2016 um 20:41
[7] `https://de.wikipedia.org/wiki/Wiki` zuletzt aufgerufen am 31.07.2016 19:22

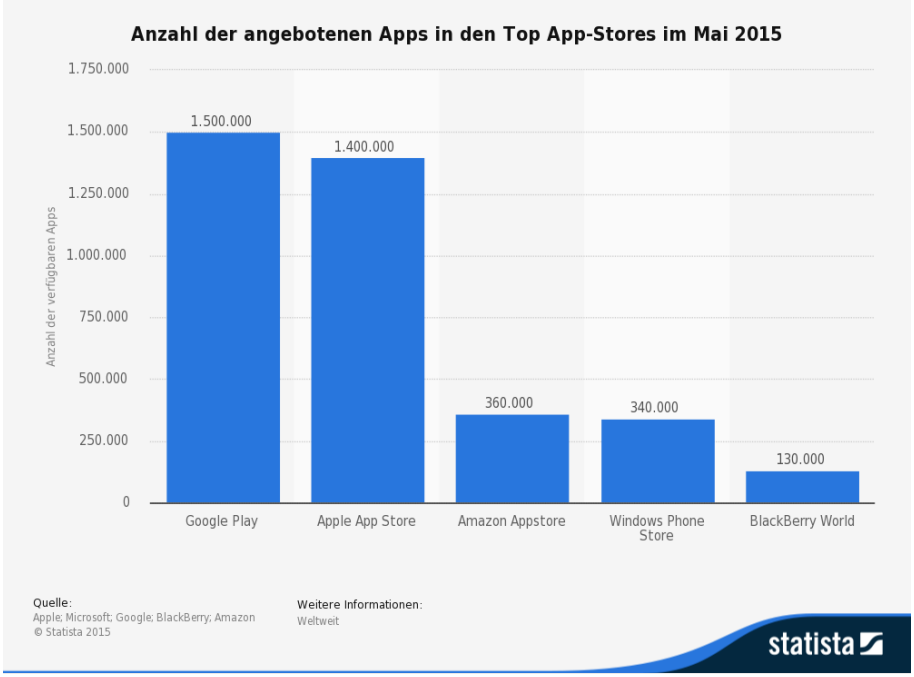

Abbildung 3: Anzahl der Applikationen in den App-Stores Mai 2015

auch starker Fokus auf die aufbereiteten Screenshots gelegt. Vermittelten diese zwei Kriterien den Eindruck, dass es sich hierbei um eine Applikation handelt, welche allgemeine Informationen aufbereitet oder um eine für Nachschlagewerke, so wurde diese ausgewählt und in weiterer Konsequenz installiert.

Bei der Auswahl der Kriterien wurden die allgemeinenen Leitfäden von Apple befolgt. Diese besagen, dass der Titel einer Applikation die Wahrnehmung der Benutzerinnen und Benutzer im App Store sehr stark beeinflusst und eine zentrale Rolle spielt. Ähnlich ist auch die Bedeutung der Screenshots beschrieben. Bei diesen heißt es, dass sie die Essenz der Applikation widerspiegeln sollen, da diese zusammen mit dem Titel als erstes dem/ der Benutzer/ Benutzerin in den Suchergebnissen angezeigt werden[8]. Abbildung 4 zeigt die angesprochenen Kriterien anhand von Suchergebnissen.

Bei der Suche wurden außerdem nur kostenfreie Applikationen berücksichtigt, da der freie Zugang zu Information eine zentrale Rolle spielt. Zusätzlich ist beschlossen worden, sich auf die ersten 20 Suchergebnisse zu beschränken. Der Grund für diese Entscheidung basiert darauf, dass mit zunehmenden Scrollen die Suchergebnisse

[8] `https://developer.apple.com/app-store/product-page/` zuletzt aufgerufen am 30.07.2015

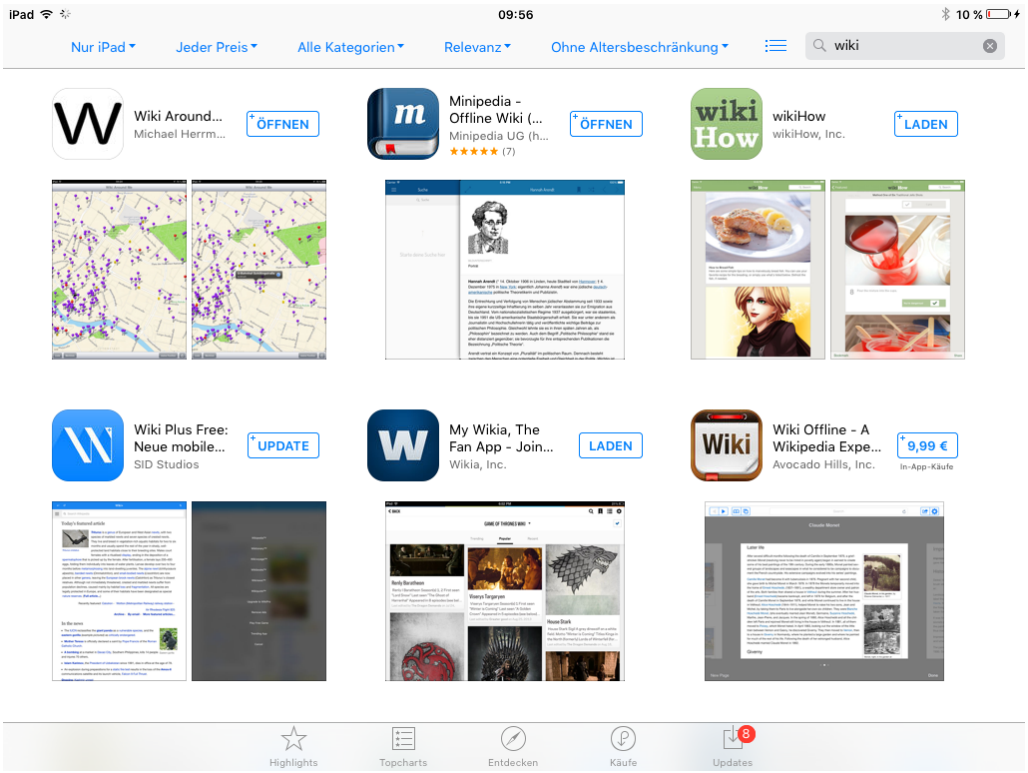

Abbildung 4: Suchergebnisse von Apple's App Store

nicht mehr passend waren. Des Weiteren wurde auch das allgemeine Verhalten der Benutzerinnen und Benutzer bedacht. Denn Personen, die beispielsweise mit Hilfe der Suchmaschine Google nach Information suchen, besuchen zu 91% nicht die zweite Seite der Suchergebnisse. 50% entscheiden sich sogar für einen Treffer aus den Top 3 der Resultate. (vgl. Sweeny, 2011)
Und da der Mensch nunmal ein Gewohnheitstier ist, so schreibt es zumindest Lenzen (2012) in ihrem Artikel über das Buch von Charles Duhigg, wurde beschlossen, dass das beschriebene Verhalten von Sweeny (2011) auf die Suchergebnisse von Google-Play-Store und Apple-Play-Store anzuwenden ist.

2.3.3 Beschreibung der ausgewählten Applikationen

Vorab ist zu erwähnen, dass lediglich jene Applikationen erwähnt und beschrieben werden, bei denen es während der Evaluierung nicht zu Abstürzen des Programms gekommen ist. Anhand der oben genannten Kriterien wurden elf Applikationen für iOS und zehn Applikationen für Android in die Evaluierung aufgenommen. Vier von

den ausgewählten Applikationen sind sowohl für iOS als auch für Android verfügbar. Im folgenden werden die ausgewählten Applikationen kurz beschrieben. Die Reihenfolge der Aufzählung dient dabei lediglich als Möglichkeit zum Referenzieren und spiegelt weder Qualität noch Wichtigkeit der jeweiligen Applikationen wider.
Um die Beschreibungen der einzelnen Applikationen besser zu verstehen, werden zuvor noch bestimmte verwendete Begriffe erklärt.

Wikipedia-Reader

Wenn in weiterer Folge von einem Wikipedia-Reader gesprochen wird, ist damit gemeint, dass die Applikation ihre Daten von der Wikipedia bezieht und diese in einem eigenen Design oder mit speziellen Skins[9] aufbereitet. Dies ist unter anderem deshalb möglich, da MediaWiki, die der Wikipedia zugrundeliegende Technologie, eine umfangreiche API bereitstellt. Mit der Funktion API:Search[10] kann beispielsweise direkt auf die Suchfunktionalität der Wikipedia zugegriffen werden. Es wird ebenfalls die Möglichkeit geboten Inhalte direkt abzurufen[11] oder auf spezielle Features zuzugreifen. Beispielsweise ist es möglich einen Zufallsartikel[12] oder Artikel in der Nähe[13] zu erhalten. Bei Applikationen mit der Bezeichnung Wikipedia-Reader wurde angenommen, dass diese Möglichkeiten zum Teil oder ganz in Anspruch genommen worden sind.

Wikipedia-Container

Als Container werden jene Applikationen bezeichnet, bei denen das eigenes Interface nur sehr marginal gehalten ist und sämtlich Inhalte direkt durch das Einbetten der Wikipedia Webseite, in einer WebView/UIWebView/WKWebView, dem/ der Benutzer/ Benutzerin präsentiert werden.

[9] `https://de.wikipedia.org/wiki/Hilfe:Skin` zuletzt aufgerufen am 13.09.2016 um 13:02
[10] `https://www.mediawiki.org/wiki/API:Search` zuletzt aufgerufen am 27.10.2016 um 19:32
[11] `https://www.mediawiki.org/wiki/Extension:TextExtracts` zuletzt aufgerufen am 13.09.2016 um 10:17
[12] `https://www.mediawiki.org/wiki/API:Search` zuletzt aufgerufen am 27.10.2016 um 19:37
[13] `https://www.mediawiki.org/wiki/API:Showing_nearby_wiki_information` zuletzt aufgerufen am 27.10.2016 um 19:40

WikiMedia Foundation, kurz: Wikimedia
Die WikiMedia Foundation, in weiterer Folge Wikimedia genannt, ist die Betreiberorganisation der Wikipedia und deren Schwesterprojekte. Zu den Schwesterprojekten zählen unter anderem Wiktionary, Wikiquote, Wikibooks uvm. [14]

2.3.3.1 Android

1. **Wiki Encyclopedia Gold**[15]
   Bei Wiki Encyclopedia Gold handelt es sich um einen Wikipedia-Reader. Die in der Applikation angebotene Suche greift dabei auf die Wissensbasis der Wikipedia zu. Neben den Inhalten der Wikipedia stellt die Applikation auch Daten anderer Wikimedia-Projekte zur Verfügung. Dazu zählen, mit aktuellem Stand, Wiktionary sowie Wikitravel, zu sehen auf Abbildung 5. Ebenfalls ist in Abbildung 5 der Start-Screen der Applikation zu erkennen. Hier werden unter anderem folgende Möglichkeiten geboten: „Artikel des Tages“ sowie „Zufälliger Artikel“. Auf „Artikel in Ihrer Nähe“ muss grundsätzlich nicht verzichtet werden. Jedoch wird immer nur ein zufälliger Artikel in der Nähe angezeigt. Eine Möglichkeit alle Artikel in der Nähe gleichzeitig, in einer nach Entfernung geordneten Liste, anzeigen zu lassen ist leider nicht möglich. Zusätzlich wird die eigentliche Entfernung, die in Abbildung 5 in rot zu sehen ist, je nach Artikelvorschau von der darunter befindlichen Werbung verdeckt. Für die Verwendung der Applikation ist generell eine Internetverbindung erforderlich. Allerdings bietet die Applikation auch eine kostenpflichtige Möglichkeit an, um Inhalte herunterzuladen und diese dann im Offline-Modus zu lesen.

2. **The Dictionary - Encyclopedia**[16]
   The Dictionary - Encyclopedia kann als eine Mischung aus Wikipedia-Container und Wikipedia-Reader gesehen werden. Denn der Start-Screen der Applikation, wie auf Abbildung 6 zu sehen ist, bietet eine Vielzahl an Abkürzungen zu gewissen Inhalten der Wikipedia, sei es durch eine direkte Suche, den Artikel des Tages oder durch einen zufälligen Artikel. Die Klassifizierung als

[14] `https://wikimediafoundation.org/wiki/Unsere_Projekte` zuletzt aufgerufen am 13.08.2016 um 11:37
[15] `https://play.google.com/store/apps/details?id=uk.co.appsunlimited.wikiapp` zuletzt aufgerufen am 11.08.2016 um 18:25
[16] `https://play.google.com/store/apps/details?id=com.hoons.wikipedia.free` zuletzt aufgerufen am 04.12.2016 um 15:26

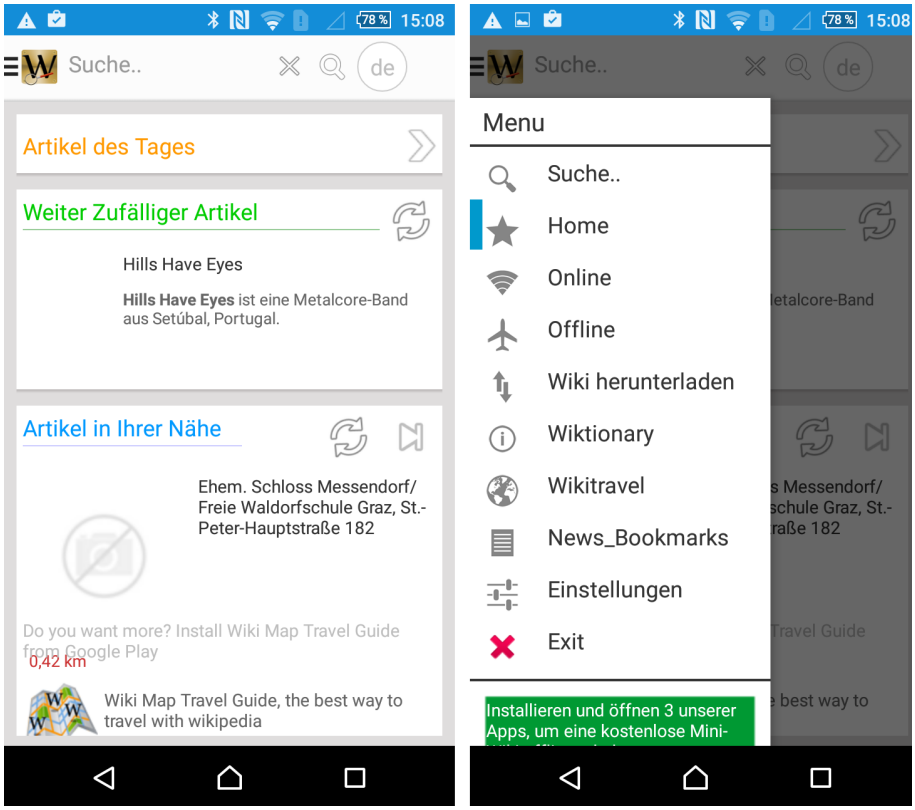

Abbildung 5: Wiki Encyclopedia Gold

Wikipedia-Container wird dadruch gerechtfertigt, dass nach einmaliger Ausführung dieser Abkürzungen das Ergebnis mit Hilfe der mobilen Wikipedia Webseite geladen wird. Diese bietet von sich aus eine Suche sowie zufällige Artikel bzw. den Start-Screen der Wikipedia, welcher unter anderem den Artikel das Tages beinhaltet, an. Im Zuge der Evaluierung hat sich herausgestellt, dass an dieser Stelle direkt die Funktionalitäten der mobilen Webseite in Anspruch genommen werden. Denn dadurch wird eine zusätzliche Interaktion erspart und wie in Kapitel 3.3.1 erwähnt wird, versuchen Benutzerinnen und Benutzer möglichst wenig Aufwand zu betreiben um an ihr Ziel zu kommen. Hier ist auch noch zu erwähnen, dass die Anzeige der mobilen Webseite der Wikipedia ebenfalls den Menüpunkt „Artikel in der Nähe“ anzeigt. Jedoch wurde die am Ende dieses Kapitel erläuterte Kommunikation zwischen Container und eingebetteten WebView entweder gar nicht oder nicht ordnungsgemäß durchgeführt, sodass diese Funktionalität nicht ausführbar ist.
Der zuvor angesprochene Teil eines Wikipedia-Readers beschreibt den Um-

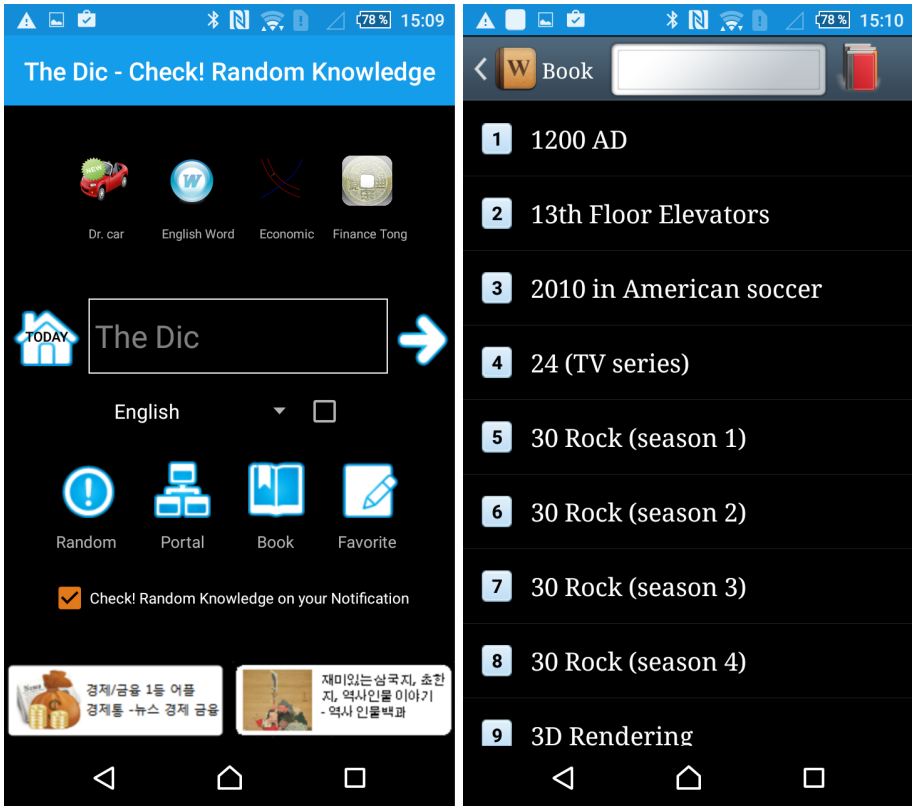

Abbildung 6: The Dictionary - Encyclopedia

stand, dass grundsätzlich die eben angesprochenen Funktionen direkt durch die Applikation ermöglicht werden. Zusätzlich zu den normalen Artikeln der Wikipedia wird auch eine Auflistung der Portale von Wikipedia:Portal[17] sowie eine Auflistung von generierten eBooks durch Wikipedia:Books[18] angezeigt. Es wird auch die Möglichkeit Benachrichtigungen zu erhalten seit kurzem angeboten. Dies hat sich nach einem Update der Applikation in der zweiten Evaluierungsphase herausgestellt. Jedoch beschränken sich diese auf zufällig ausgewählte Artikel und haben keine Relevanz zum/ zur derzeitigen Benutzer/ Benutzerin. Diese Umstände lassen im Endeffekt die Beurteilung als Teil-Wikipedia-Reader zu.

[17] `https://de.wikipedia.org/wiki/Wikipedia:Portale` zuletzt aufgerufen am 02.12.2016 um 18:33

[18] `https://en.wikipedia.org/wiki/Wikipedia:Books` zuletzt aufgerufen am 02.12.2016 um 18:34

3. **GWiki - Wikipedia for Android**[19]

   GWiki - Wikipedia for Android ist ein Wikipedia-Container. Wie in Abbildung 7 zu sehen, bietet diese Applikation einige Abkürzungen, um direkt neue Informationen abzurufen. Einmal ausgeführt, wird das Ergebnis in einer WebView geladen, welches die mobile Webseite der Wikipedia anzeigt. An dieser Stelle sind ebenfalls alle Funktionen der mobilen Webseite ausführbar. Die grundsätzlich übersichtlichen und schnell auszuführenden Abkürzungen rücken bei manueller, linkgestützter Navigation leider in weite Ferne. Denn für jeden Link, der so geöffnet wird, muss einmal der Hardware Back-Button gedrückt werden um schlussendlich wieder zur Übersicht der angebotenen Funktionen zu gelangen. GWiki ist nur mit aktiver Internetverbindung benutzbar.

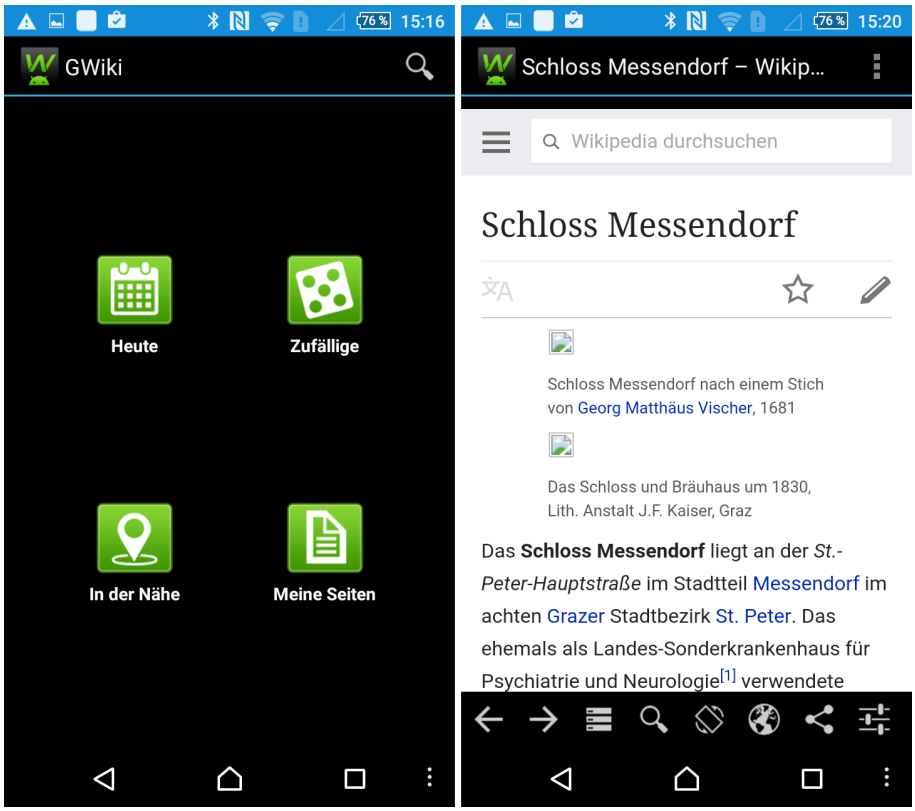

Abbildung 7: GWiki - Wikipedia for Android

[19] `https://play.google.com/store/apps/details?id=org.strive.wikipedia` zuletzt aufgerufen am 11.08.2016 um 19:09

4. **EveryWiki: Wikipedia++**[20]
   EveryWiki ist ein Wikipedia-Reader, der nicht nur Inhalte der Wikipedia und deren Schwesterprojekte präsentiert, sondern auch auf zahlreiche andere Wikis zugreift. So sind beispielsweise Wikis zu Filmen, Videospielen, Büchern oder Sammlungen mit satirischen Inhalten zu finden. Besteht eine aktive Internetverbindung, ist es möglich in einem breit gefächerten Angebot zu stöbern, welches aus einer Vielfalt von unterschiedlichen Informationen zusammengesetzt ist. Zusätzlich dazu gibt es in jeder einzelnen auszuwählenden Rubrik die Zufallsfunktion. Abbildung 8 zeigt links den Aufbau eines gewählten Wikis sowie Beispiele von verfügbaren Wikis auf der rechten Seite. Die Zufallsfunktion ist dabei nicht zentral von einer Stelle aus erreichbar. Um diese erneut auszuführen muss entweder mit Hilfe des Back-Buttons wieder zurück zur Hauptseite navigiert werden oder das ausgewählte Wiki über das Menü erneut aufgerufen werden. Wird ein zufälliger Artikel angefordert, dann wird zunächst eine Auswahl an zufälligen Artikeln angezeigt. Aus dieser erstellten Liste können Benutzerinnen und Benutzer anschließend einen bestimmten Artikel auswählen. Zusätzlich präsentiert die Applikation die Inhalte der jeweiligen Wikis nicht nur mit Hilfe der jeweiligen geladenen Webseiten, sondern stellt diese in einem eigenen Skin da. Grafisch zwar angepasst, wird dennoch die Möglichkeit geboten auf die von den jeweiligen Hauptseiten zur Verfügung gestellen Funktionen zurückzugreifen. Ist beispielsweise die Hauptseite der Wikipedia aktiv, können deren Funktionalitäten wie „Artikel des Tages“ oder „In den Nachrichten“ genutzt werden. Artikel in der unmittelbaren Nähe oder basierend auf einer Position sowie eine Möglichkeit Inhalte Offline zu lesen wird hier nicht geboten.

5. **Random Wiki - learn new things**[21]
   Random Wiki - learn new things (Abbildung 9) ist ein weiterer Wikipedia-Container dessen Hauptaugenmerk auf die Präsentation von zufälligen Artikeln gerichtet ist. Dies wird mit einem zentralen Button in der ActionBar[22] un-

[20] `https://play.google.com/store/apps/details?id=net.nebulium.wiki` zuletzt aufgerufen am 11.08.2016 um 21:23

[21] `https://play.google.com/store/apps/details?id=co.sharan.random2wiki` zuletzt aufgerufen am 12.08.2016 um 9:48

[22] `https://developer.android.com/design/patterns/actionbar.html` zuletzt aufgerufen am 02.12.2016 um 18:25

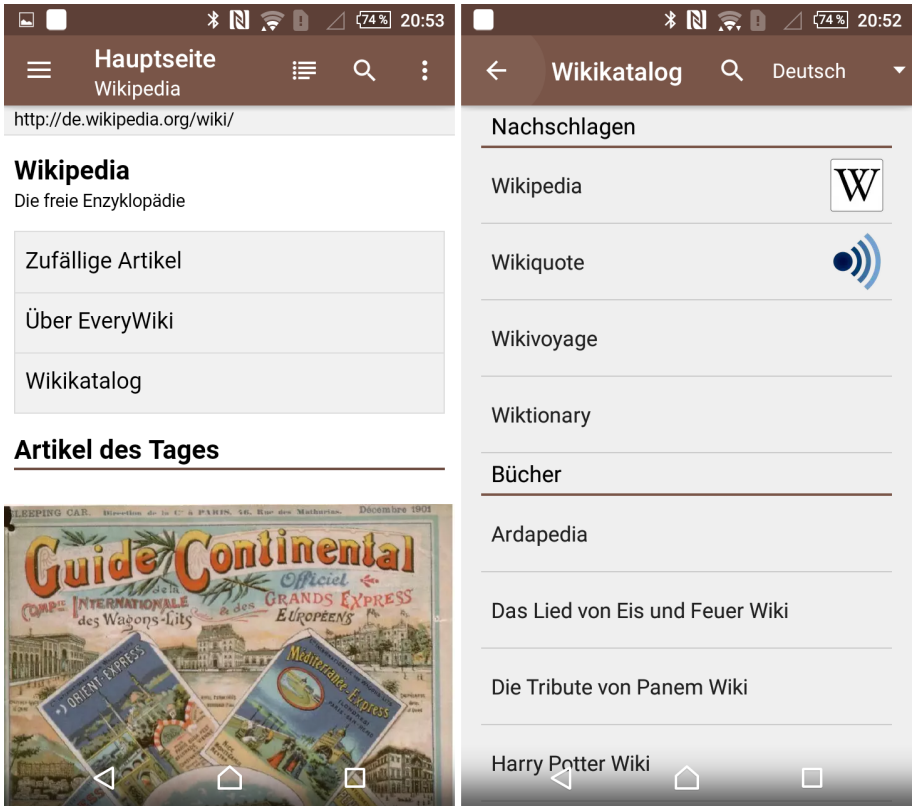

Abbildung 8: EveryWiki: Wikipedia++

termauert. Da es sich hier um einen Wikipedia-Container handelt, ist ebenfalls der Zufallszugriff über das Menü der mobilen Wikipedia Webseite möglich. Es wird zwar in diesem Kontext auch die Möglichkeit des „Suchens in der Nähe“ angeboten, jedoch funktioniert diese aus dem am Ende dieses Kapitels beschriebenen Problems der fehlenden Berechtigung nicht. Für die Nutzung ist die Verbindung zum Internet über mobile Netzwerke oder WLAN erforderlich.

6. **NearbyWikipedia**[23]

   NearbyWikipedia dient als Container um Artikel in der unmittelbaren Umgebung oder eines bestimmten Ortes anzuzeigen. Artikel in der Nähe werden hier anhand des aktuellen Standpunktes gefiltert. Besteht das Interesse relevante Artikel, ausgehend von einem bestimmten Ort, zu erkunden, wird die Möglichkeit geboten entweder nach einer gewünschten Adresse zu suchen oder

[23] `https://play.google.com/store/apps/details?id=pkg.NearbyWikipedia` zuletzt aufgerufen am 12.08.2016 um 09:58

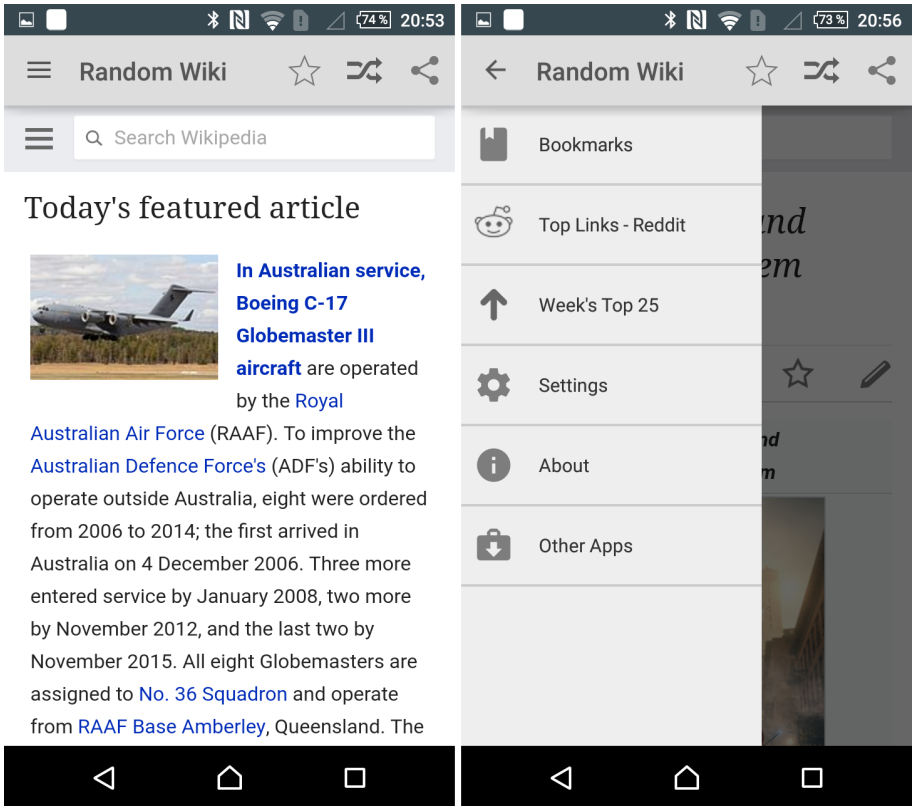

Abbildung 9: Random Wiki - learn new things

durch manuelles Navigieren auf der angezeigten Landkarte einen Ausschnitt auszuwählen. Wurde eine Adresse bzw. ein Ausschnitt festgelegt, so werden anhand dieser Information Artikel geladen und als Pins auf der Landkarte platziert, wie in Abbildung 10 auf der linken Seite zu sehen ist. Interessiert sich ein/ eine Benutzer/ Benutzerin für eine bestimmte Information, so kann dieser/ diese auf einen der Pins drücken. Der durch den Pin dargestellte Artikel wird dann mit Hilfe der mobilen Webseite der Wikipedia, welche in einer WebView eingebettet ist, geladen. Grundsätzlich sind in weiterer Folge auch die angebotenen Funktionen der mobilen Webseite verfügbar. Interessant ist hier allerdings, dass die Funktion „Artikel in der Nähe “ nicht ausführbar ist. Grund dafür ist die mangelnde oder nicht ordnungsgemäße Kommunikation zwischen WebView und Container. An dieser Stelle wird wieder auf das Ende dieses Kapitels verwiesen, wo dieses Problem disktutiert wird.

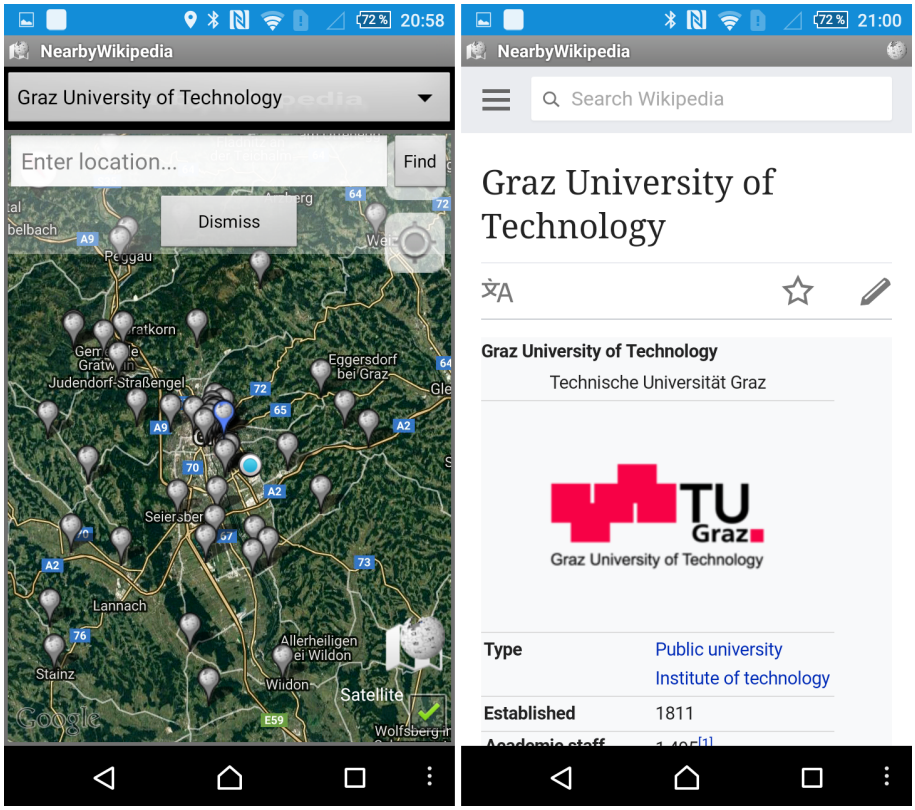

Abbildung 10: NearbyWikipedia

2.3.3.2 iOS

7. **Wiki Around Me**[24]

   Wiki Around Me ist ähnlich aufgebaut wie „NearbyWikipedia“ aus dem Play-Store und ist ein weiterer Wikipedia-Container, der darauf abzielt Artikel in der Nähe zu präsentieren. Realisiert wird dies wiederum durch das Anzeigen einer Landkarte, auf der anschließend Pins positioniert werden. Abbildung 11 zeigt ein Beispiel einer solchen Landkarte. Zusätzlich gibt es auch die Möglichkeit eine Liste anzeigen zu lassen, auf der in der Nähe befindliche Artikel gelistet sind. Bei Betätigung des Aktualisierungssymbols werden je nach aktuellem Kartenausschnitt neue Pins geladen. Dabei ist es auch möglich, dass bei gleichbleibendem Kartenausschnitt unterschiedliche Pins platziert werden. Die Liste hingegen zeigt unabhängig von der Karte immer dieselben Artikel an. Aufgrund der Listeneinträge und deren angegebenen Entfernungen wird

[24] `https://itunes.apple.com/at/app/wiki-around-me/id452992419?mt=8` zuletzt aufgerufen am 12.08.2016 um 13:35

vermutet, dass es sich hierbei um Artikel handelt, die ausgehend von der eigentlichen Position des Gerätes berechnet werden. Artikel in der Nähe von einem bestimmten Ort anzeigen zu lassen ist in dieser Applikation nicht ohne gewissen Aufwand möglich. Da keine Suche angeboten wird, muss händisch zum passenden Kartenausschnitt navigiert werden. Wenn sich Benutzerinnen

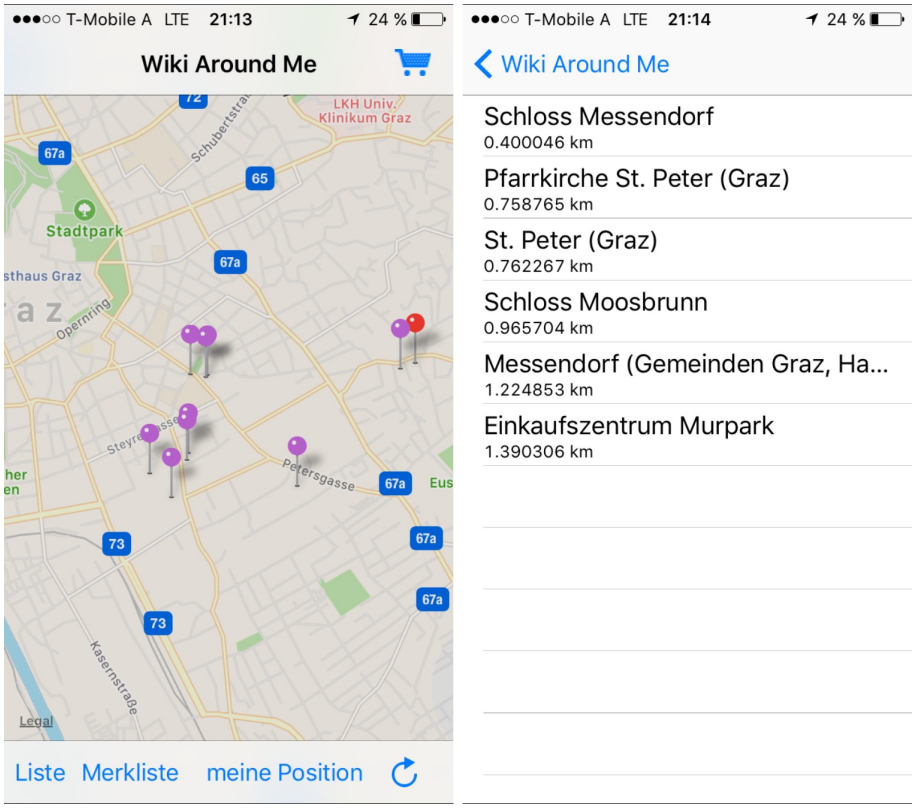

Abbildung 11: Wiki Around Me

und Benutzer für einen Pin entscheiden und diesen als wissenswert einstufen, können sie mehr über diesen, durch Berührung, in Erfahrung bringen. Dabei wird der ausgewählte Wikipediaartikel in einem Container geladen. Dieser wird direkt durch die Anzeige der Desktopversion der Webseite präsentiert. Bei genauerer Betrachtung der Desktopversion stellt sich heraus, dass auch auf die mobile Version gewechselt werden kann. Diese bietet dann die bekannten Funktionen der mobilen Webseite. Da Wiki Around Me die Inhalte direkt nach Badarf lädt, ist für die Verwendung eine aktive Internetverbindung vonnöten.

8. **Articles**[25]

   Articles ist ein Wikipedia-Reader, bei dem die angezeigten Artikel in einem eigenen aufbereiteten Design wiedergegeben werden. Ob im Hintergrund ein eigener Skin für die Anzeige verwendet wird oder die Inhalte der Artikel direkt per API Zugriff abgefragt werden, kann an dieser Stelle nicht festgestellt werden. Es wird auch die Möglichkeit geboten, ähnlich wie bei einem normalen Browser, dass mehrere Tabs gleichzeitig geöffnet sein können und ein Wechsel zwischen diesen möglich ist. Grundsätzlich gibt es auch eine „In der Nähe“ und „Überrasche mich!“ Funktion. Für die Umsetzung der Ersteren wird mit einer Landkarte und Pins gearbeitet. Die Handhabung erfolgt dabei gleich wie bei den zuvor erläuterten Applikationen, welche für die Darstellung von Artikel in der Nähe eine Landkarte verwenden. Zweitere ist für einen zufälligen Artikel verantwortlich. Jedoch muss auch erwähnt werden, dass diese Funktionen nicht gleich ersichtlich sind. Sie befinden sich nämlich in der Rubrik „Lesezeichen“, wie in Abbildung 12 zu sehen ist. Wenn Benutzerinnen und Benutzer keine Lesezeichen setzen und dadurch diese Ansicht niemals öffnen, könnte ihnen diese Möglichkeit entgehen. Um neue Artikel zu lesen muss in erster Instanz eine aktive Verbindung zum Internet bestehen. Wurde ein Artikel allerdings bereits geladen, so wird dieser automatisch auf dem Gerät gespeichert, sodass bei späterer Offline-Nutzung diese ebenfalls lesbar sind. Im Grunde sind dadurch alle einmal aufgerufenen Artikel auch ohne Internetverbindung verfügbar und abrufbar.

   Ob ausnahmslos alle aufgerufenen Artikel lokal gespeichert werden, kann an dieser Stelle nicht gesagt werden. Grundsätzlich wird von einem internen Limit ausgegangen, sei es die Begrenzung durch den Speicherplatz am Gerät oder durch andere etwaige Limitierungen. Diese Grenze wurde aber nicht ausgereitzt. De facto wurden 25 Artikel getestet. Diese konnten nach einmaligem Aufrufen in anderen Tabs, mit aktivem Flugmodus, nachträglich erneut angezeigt werden.

[25] `https://itunes.apple.com/us/app/articles/id364881979?mt=8` zuletzt aufgerufen am 12.08.2016 um 13:49

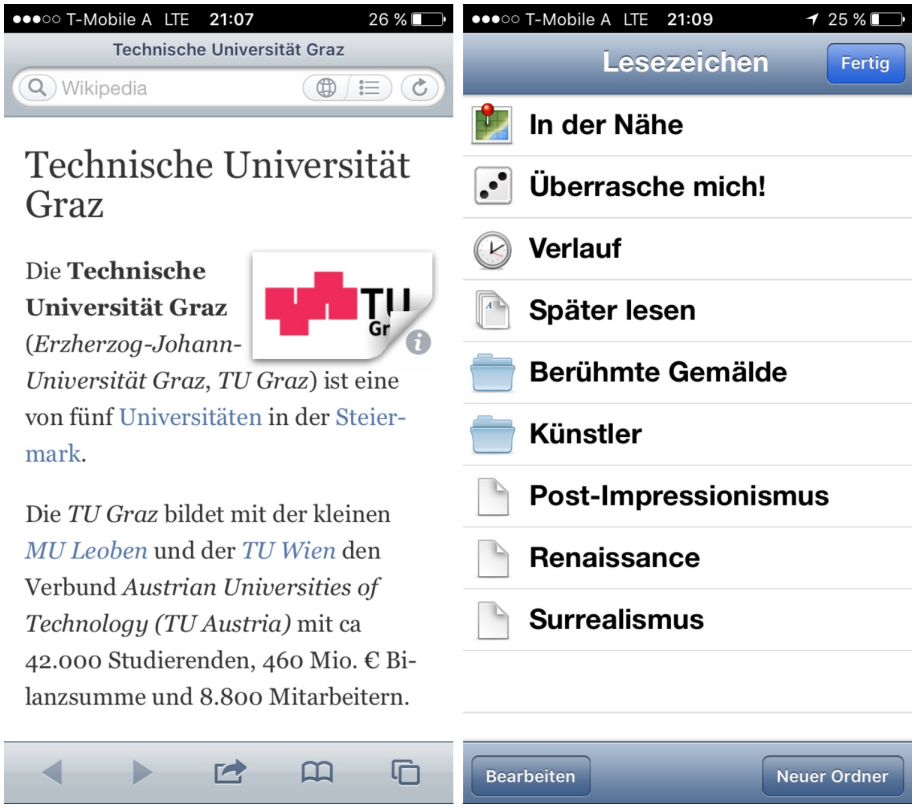

Abbildung 12: Articles

9. **WikiExplorer for Wikipedia**[26]

   Wie der Name der Applikation erahnen lässt, geht es hier um einen Wikipedia-Reader, der durch exploratives Erkunden aufgrund der aktuellen Position auf Artikel in der Nähe aufmerksam macht. Angezeigt werden Artikel mit Hilfe einer Landkarte. Das Prozedere dabei ist wie folgt: Benutzerinnen und Benutzer müssen durch Navigieren oder durch Lokalisierung der eigenen Position einen Kartenausschnitt auswählen und führen dann in weiterer Folge einen sogenannten „Long Press“ auf der Karte aus. Ausgehend vom berührten Punkt wird ein Radius gezogen. Innerhalb dieses Radius werden Pins, welche auf Artikel hinweisen, auf der Landkarte platziert. Durch Berührung auf einen dieser Pins kann der dazugehörige Artikel gelesen werden. Abbildung 13 zeigt solch einen Radius mit platzierten Pins. Außerdem ist auch die Darstellung eines Wikipedia Artikels zu sehen. Ausgehend von dem Design, in dem der

[26] `https://itunes.apple.com/at/app/wikiexplorer-for-wikipedia/id1023141634?mt=8` zuletzt aufgerufen am 12.08.2016 um 15:30

Artikel dargestellt wird, ist es klar, dass hier die Standartanzeige eines Wikipedias Artikels nicht zum Tragen kommt. Aufgrund dieser Tatsache ist es eindeutig, dass diese Applikation als Wikipedia-Reader einzustufen ist. Besteht allerdings ein Interesse daran zu erfahren, welche Artikel es in der Nähe von einem bestimmten Ort oder Adresse gibt, dann muss dafür ein wenig Aufwand betrieben werden. Es gibt leider keine Suche nach bestimmten Orten. Deshalb muss mit Hilfe von Gesten gearbeitet werden und manuell der Kartenausschnitt des gewünschten Ortes gesucht werden. Für den WikiExplorer ist eine aktive Internetverbindung nötig um die Applikation zu nutzen.

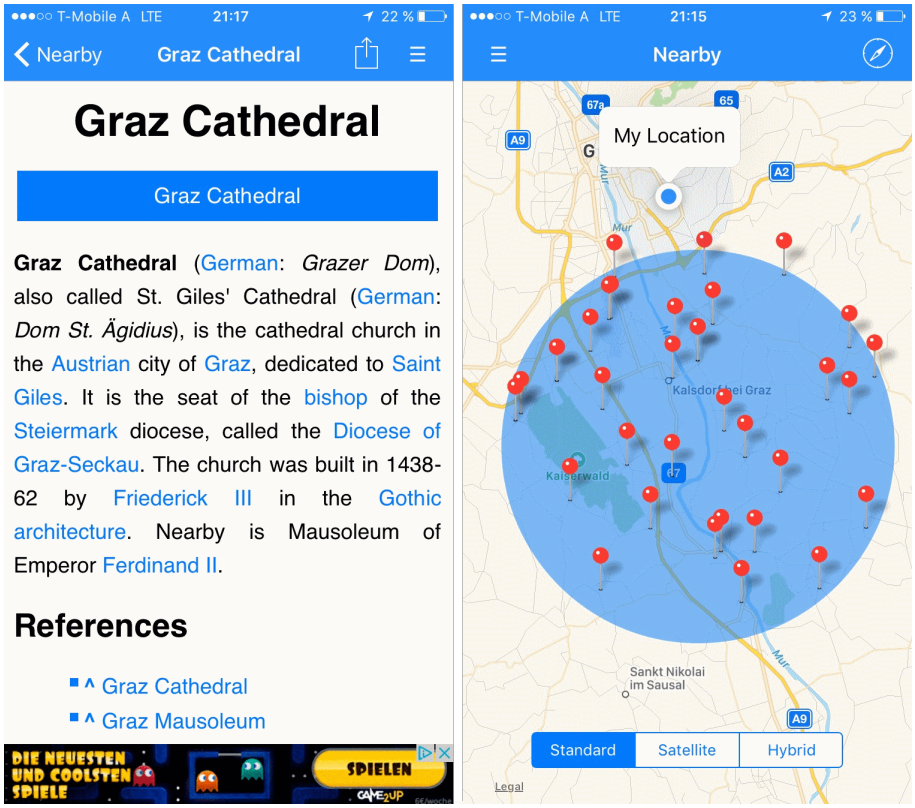

Abbildung 13: WikiExplorer for Wikipedia

10. **Wikipanion**[27]

    Wikipanion, zu sehen in Abbildung 14, bedient sich seiner Inhalte ebenfalls der Wikipedia, präsentiert diese aber nicht nur durch Abbildung der Webseite

[27] `https://itunes.apple.com/at/app/wikipanion/id288349436?mt=8` zuletzt aufgerufen am 12.08.2016 um 15:46

der Wikipedia in einer WebView. Daher ist diese Applikation als Wikipedia-Reader zu bezeichen. Neben einer normalen Suche werden auch Funktionen wie „Zufälliger Artikel“ und „In Nähe der akt. Position“ angeboten. Um Artikel

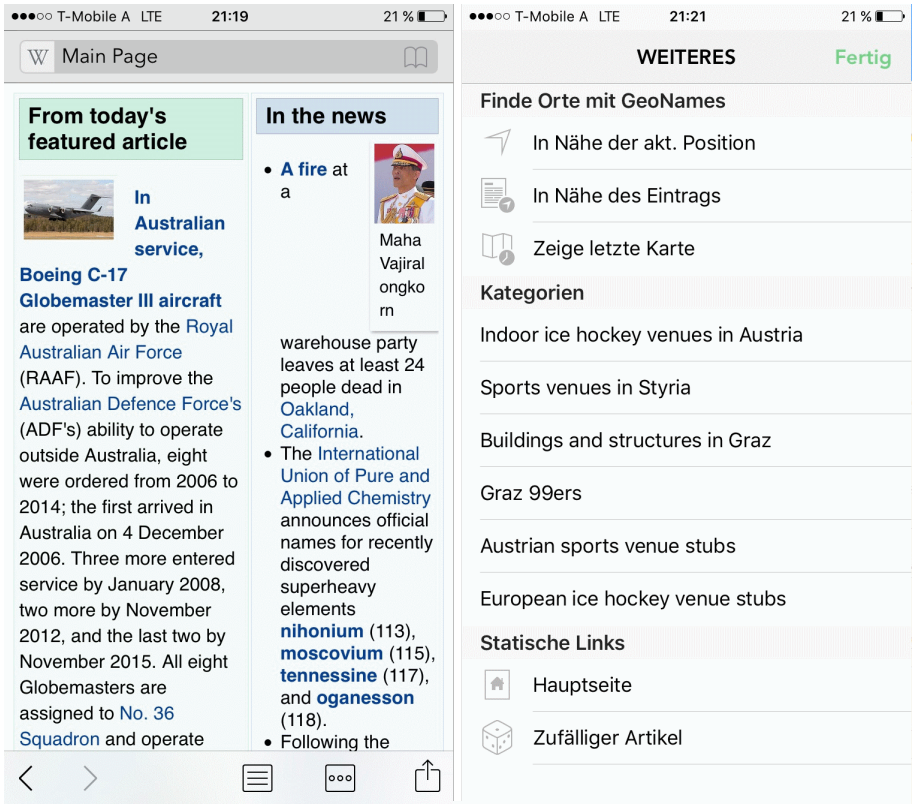

Abbildung 14: Wikipanion

in der Nähe zu erforschen bietet Wikipanion zwei Wege. Der eine ermöglicht die Anzeige von Artikeln auf einer Landkarte mit Hilfe von Pins. Das Lesen eines solchen Artikel ist wieder durch Berührung möglich. Als zweiter Weg wird hier auch die Möglichkeit geboten die Artikel in der Nähe in einer geordneten Liste anzeigen zu lassen. Es wird ebenfalls die Möglichkeit geboten interessante Beiträge in der Nähe eines aktuellen Artikels anzeigen zu lassen. Dies ist allerdings nur dann möglich wenn, der aktuelle Artikel mit Koordinaten versehen ist. Wird auf diese Funktion zurückgegriffen, werden wieder beide Visualisierungsarten angeboten. Dank letzteren wird Benutzerinnen und Benutzer folgendes Szenario eröffnet: Lassen Benutzerinnen und Benutzer sich durch einen zufälligen Artikel überraschen und ist dieser mit Koordinaten versehen können sie schnell in Erfahrung bringen, sofern von Interesse, was für

spannende Sachen es noch in der Nähe von diesem gibt.

11. **Minipedia – Offline Wiki (Wikipedia Reader)**[28]
    Minipedia - Offline Wiki ist ein Wikipedia-Reader, der Inhalte auch offline zur Verfügung stellt. Das Prozedere ist dabei wie folgt: Nach Installation und anschließendem Starten der Applikation wird angezeigt, dass für die Nutzung erst Inhalte heruntergeladen werden müssen. Auf Abbildung 15 ist dabei zu sehen, dass Pakete in verschiedenen Größen angeboten werden. Ebenfalls erkennbar ist, dass lediglich zwei kleinere Pakete, ohne dafür zu bezahlen, erhältlich sind. Wurde eines dieser ausgewählt und geladen, bietet die Applikation die Möglichkeit des „Zufälligen Artikels“ sowie die Anzeige von Artikel “In der Nähe“. Im Zuge der Evaluierung wurde versucht die “In der Nähe“ Funkti-

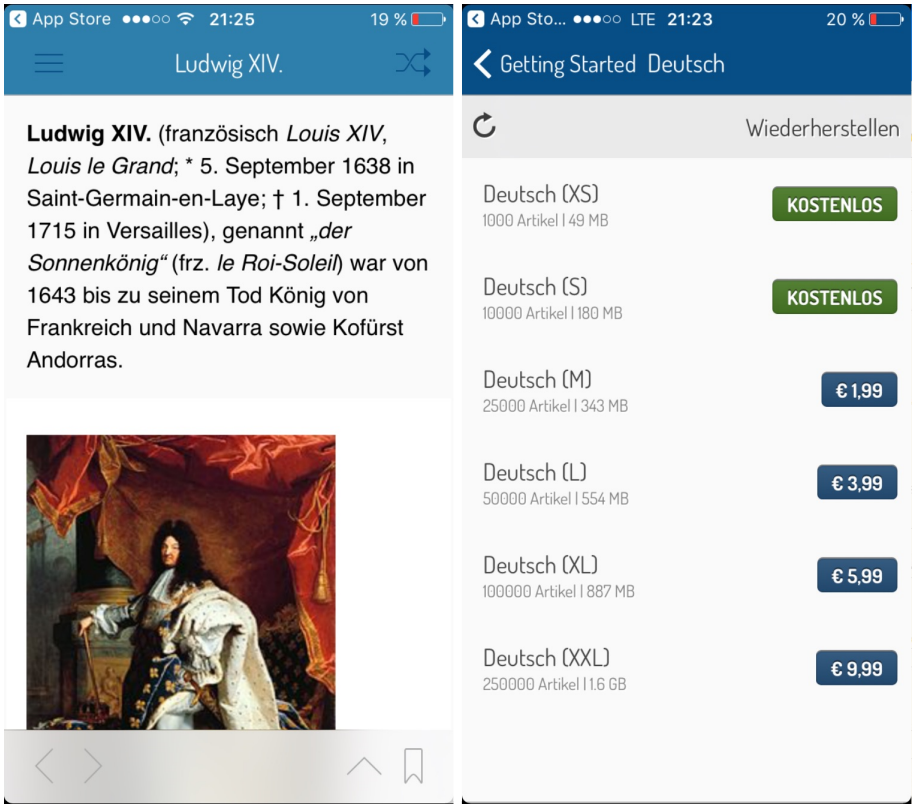

Abbildung 15: Minipedia – Offline Wiki (Wikipedia Reader)

on zu testen. Bei Ausführung der Funktionalität wurde zwar eine Landkarte

[28] `https://itunes.apple.com/de/app/minipedia-offline-wiki-wikipedia/id473512078?mt=8` zuletzt aufgerufen am 12.08.2016 um 16:07

geladen, jedoch fehlten weitere Informationen. Artikel in der Nähe wurden keine angezeigt. Ob dieses Problem verbunden mit kleineren gratis Paketen ist, da in diesem Fall nicht viele Artikel zur Verfügung stehen, oder ob ein genereller Fehler im Programm dafür verantwortlich ist, kann an dieser Stelle nicht beantwortet werden. Auf Grund des Aufbaus des Layouts kann aber grundsätzlich die Annahme getroffen werden, dass es sich um ähnliches, schon beschriebenes Verhalten handelt wie bei anderen Applikationen, die Artikel in der Nähe mit Hilfe einer Landkarte anzeigen.
Die in der Applikation angebotene textbasierte Suche ist nicht nur auf das lokale Artikelverzeichnis beschränkt. Vielmehr werden auch andere größere Pakete durchsucht. Wird ein Suchergebnis ausgewählt, welches nicht in dem heruntergeladenen Paket zur Verfügung steht, wird ein dementsprechender Hinweis angezeigt. Dieser besagt, dass der Artikel nur in einem größeren Paket verfügbar ist und dieses herunterladbar ist. Für größere Pakete ist aber extra zu zahlen. Generell ist für die Verwendung der Applikation nur bei Erstinstallation, oder wenn mehr Inhalte benötigt werden, eine Internetverbindung erforderlich. Wurden diese bereits geladen, so ist es möglich alle Artikel auch offline zu lesen ohne etwaige Einschränkungen. Dementsprechend ist auch die Zufallsfunktion ohne Einschränkung nutzbar.

12. **Wiki Plus Free: Neue mobile Lesen und Browser Tool**[29]
Mit Wiki Plus wird Benutzerinnen und Benutzer ein klassischer Wikipedia-Container geboten. Wie auf Abbildung 16 zu sehen ist, wird hier lediglich die mobile Webseite der Wikipedia in einer eingebetteten WebView angezeigt. Der Container selbst bietet einen „Home-Button“ an, bei dem nach Betätigung die Hauptseite der Wikipedia geladen wird. Dank der eingebetteten mobilen Webseite der Wikipedia ist es auch möglich zufällige Artikel zu erhalten oder nach diesen zu suchen. Artikel in der Nähe können aber nicht angezeigt werden. Die Ursache dafür wird am Ende dieses Kapitel etwas genauer beschrieben, wenn auf diese Thematik im Kontext von iOS darauf eingegangen wird. Neben Inhalten der Wikipedia wird auch die Möglichkeit geboten andere Startseiten der Wikimedia innerhalb der eingebetteten WebView zu laden. Wie in Abbildung 16 ebenfalls zu sehen ist, kann auch Wiktionary, Wikibooks und uvm. gela-

[29] `https://itunes.apple.com/at/app/wiki-plus-kostenlos-ihr-neues/id670522549?mt=8` zuletzt aufgerufen am 05.10.2016 um 20:33

den werden. Damit die Inhalte geladen werden können, bedarf es einer aktiven Verbindung zum Internet.

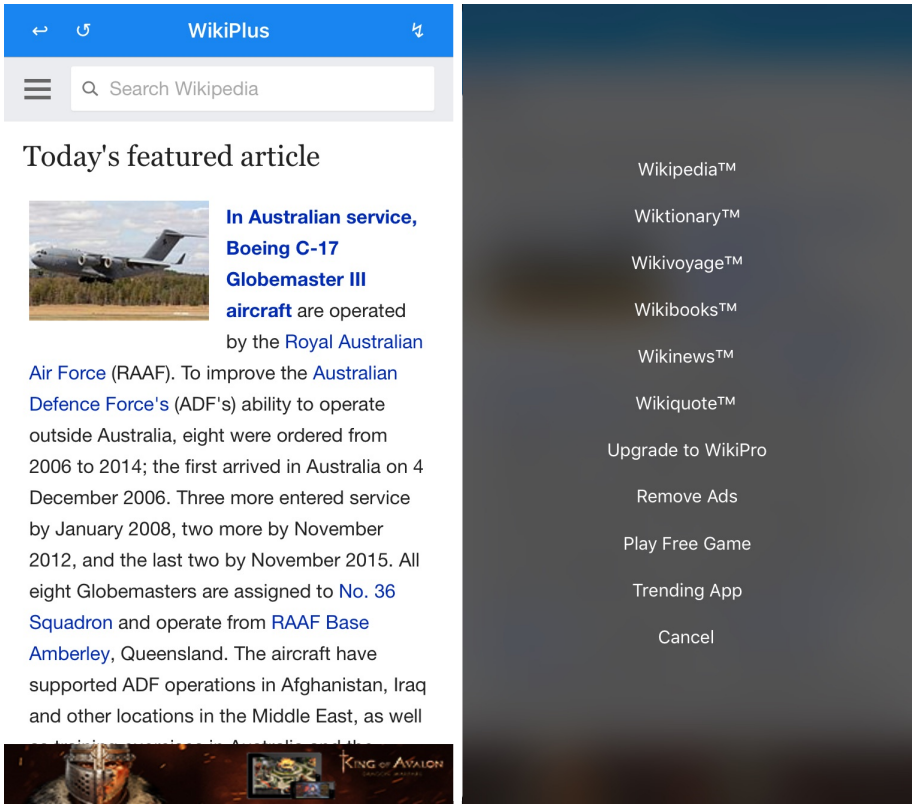

Abbildung 16: Wiki Plus Free: Neue mobile Lesen und Browser Tool

13. **Encyclopædia Britannica**[30]

    Wie durch den Namen der Applikation schon erkennbar ist, handelt es sich hierbei um die offizielle Applikation der Encyclopedia Britannica[31]. Funktionen, die zufällige Artikel liefern oder Artikel in der unmittelbaren Nähe anzeigen und so den Wissensdurst von unentschlossenen Benutzerinnen und Benutzer stillen, gibt es hier leider nicht. Dafür wird unter anderem die Rubrik „This Day“ angeboten. In dieser wird angezeigt, was an dem aktuellen Tag sowie an dem aktuellen Datum in der Vergangenheit, Interessantes geschehen ist. Neben der zuvor erwähnten Rubrik wird auch die Möglichkeit geboten, Inhalte und Wissen spielerisch kennenzulernen. Dies geschieht mit Hilfe von

[30] `https://itunes.apple.com/us/app/encyclop-dia-britannica/id447919187?mt=8` zuletzt aufgerufen am 12.08.2016 um 18:23

[31] `https://www.britannica.com/` zuletzt aufgerufen am 02.12.2016 um 18:28

generierten Quizzes. Die Gliederung beinhaltet eine Handvoll von Kategorien, welche in sich noch weiter aufgeteilt sind. Dementsprechend gibt es eine Vielzahl an unterschiedlichen Themen. Ausgehend von diesen wird ein Quiz erstellt. In Abbildung 17 sind auf der rechten Seite einige der übergeordneten Kategorien zu sehen. Zusätzlich wird eine Suche und das Stöbern im kompletten Archiv, welches alphabetisch geordnet ist, angeboten.
Erwähnenswert ist ebenfalls, dass ausgehend von einem Artikel eine Linkmap erzeugt werden kann, in der alle ausgehenden Links visualisiert werden. Wird einer dieser ausgehenden Links gedrückt, dann werden weitere ausgehende Links von diesem Artikel nachgeladen. So ensteht nach und nach ein Netz von verbundenen Artikeln, bei dem gut zu erkennen ist, wie die einzelnen Artikel miteinander vernetzt sind. Abbildung 17 zeigt auf der linken Seite solch eine erstellte Linkmap. Um die Applikation aktiv nutzen zu können muss eine Verbindung zum Internet bestehen.

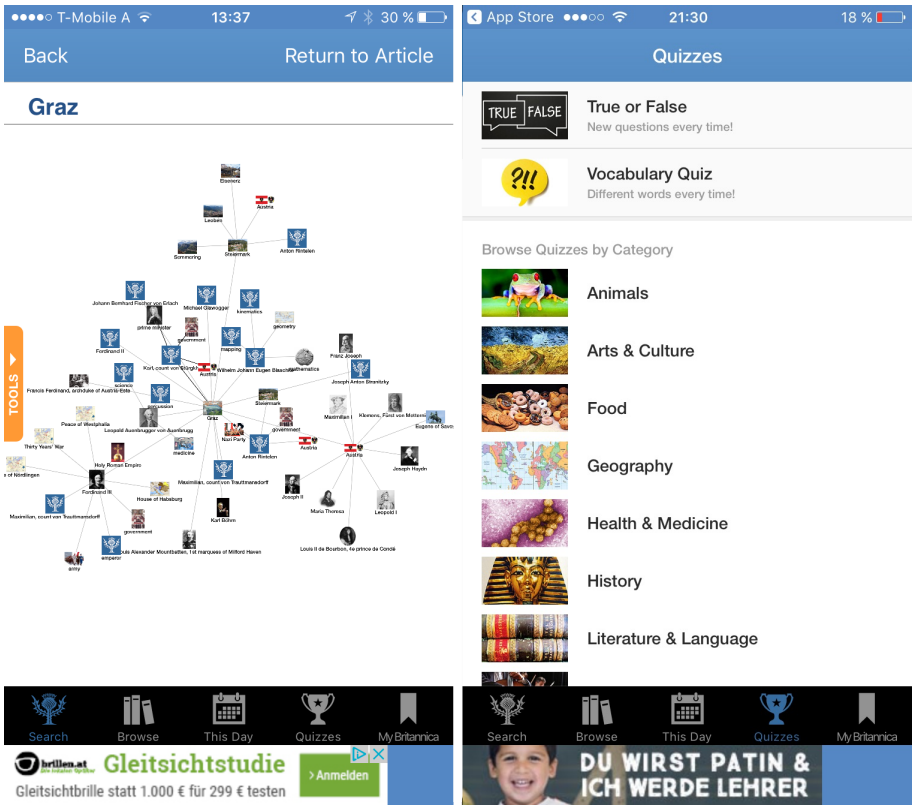

Abbildung 17: Encyclopædia Britannica

2.3.3.3 iOS / Android

14. **Kiwix**[32]

    Kiwix kann zwar im engeren Sinne als Wikipedia-Reader bezeichnet werden, ist aber im Grunde nicht nur auf Inhalte der Wikipedia beschränkt. Ähnlich wie bei Minipedia müssen Benutzerinnen und Benutzer, bevor sie das erstemal Inhalte lesen können, etwaige Pakete herunterladen. Diese gibt es ebenfalls in vielen Sprachen und in unterschiedlichen Größen. Die angebotenen Inhalte sind aber alle kostenfrei, da die Inhalte hauptsächlich aus freien Wikis stammen. Abbildung 18 zeigt links die iOS-Version der Applikation sowie Beispiele von angebotenen Paketen. Hier ist außerdem zu erkennen, dass neben Inhalten der Wikipedia und der Wikimedia Foundation auch andere Inhalte bereitgestellt sind. Rechts, auf Abbildung 18, ist die Android-Version zu sehen. Diese un-

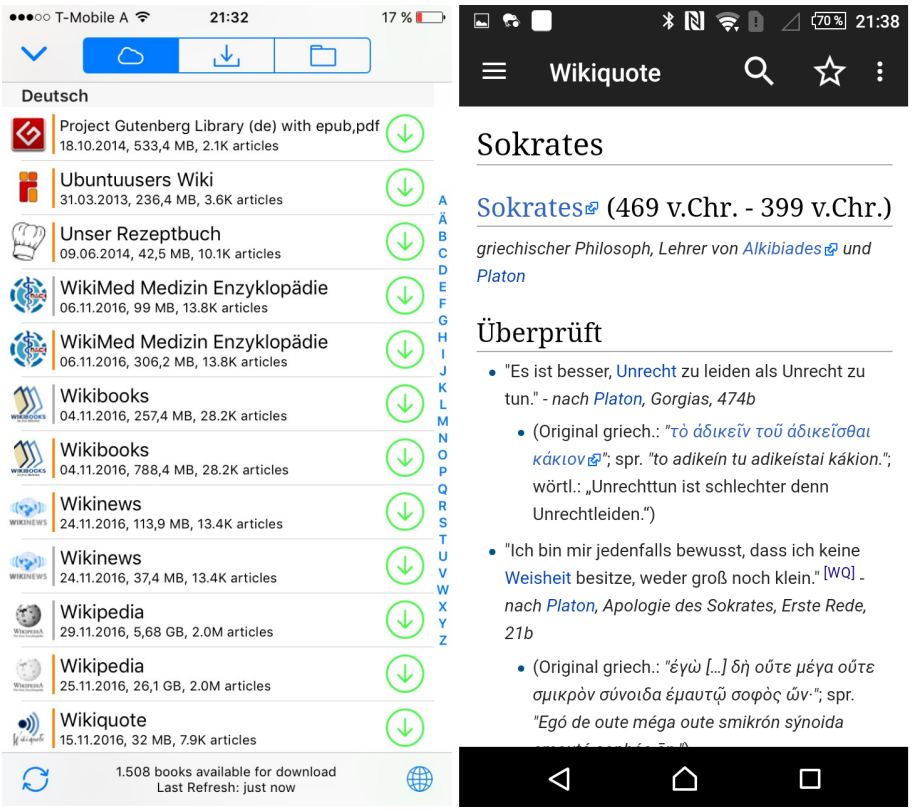

Abbildung 18: Kiwix für iOS / Android

[32] `https://itunes.apple.com/at/app/kiwix/id997079563?mt=8` `https://play.google.com/store/apps/details?id=org.kiwix.kiwixmobile&hl=de` jeweils zuletzt aufgerufen am 12.08.2016 um 18:26

terscheidet sich von der iOS mit aktuellem Stand vor allem darin, dass es hier möglich ist ZIM-Dateien[33] zu importieren. Wenn Daten, Artikel oder sonstige Informationen in diesem Format bereitliegen, können diese ohne Probleme importiert werden. Es ist durchaus auch möglich diese Daten anderswo zu laden und anschließend auf das Gerät zu übertragen. Die Applikation durchsucht darauf hin das Dateisystem auf Dateien im ZIM-Format und schlägt vor diese in die Applikation zu importieren.
Als weiterer Unterschied ist die Funktion des zufälligen Artikels anzumerken. Denn diese ist mit aktuellem Stand der Android-Version vorbehalten. Bei Kiwix für iOS muss auf diese verzichtet werden. Der zufällige Artikel wird dabei aus der aktuell ausgewählten Datenbank bestimmt. Auf die Möglichkeit Artikel in der Nähe anzuzeigen muss hier leider generell verzichtet werden. Dafür bietet die Applikation die Möglichkeit riesige Mengen an Daten offline zugänglich zu machen.

15. **Enzyklopädie / Encyclopedia**[34]
    Bei den zwei hier genannten Applikationen handelt es sich um Nachschlagewerke für Begriffe. Unterschiede finden sich in der Sprache, für die sie implementiert sind. Ein Wort bzw. ein Begriff wird hier in vielen verschiedenen Sprachlisten nachgeschlagen und anschließend dafür relevante Suchergebnisse präsentiert. Abbildung 19 zeigt ein Beispiel anhand eines gesuchten Begriffes. Hier handelt es sich rein um ein Nachschlagewerk, die Faktoren des Erkundens und des explorativen Kennenlernens von neuem Wissen sind hier nicht gegeben.

16. **Wikipedia**[35]
    Bei Wikipedia handelt es sich um die offizielle mobile Applikation der Wikipedia für Android und iOS. Einzustufen ist diese als Wikipedia-Reader,

[33] `https://de.wikipedia.org/wiki/ZIM_(Datenformat)` zuletzt aufgerufen am 02.12.2016 um 18:29

[34] `https://play.google.com/store/apps/details?id=nl.dirkslot.encyclo.de` `https://itunes.apple.com/de/app/enzyklopadie-de/id964502204?mt=8` `https://play.google.com/store/apps/details?id=nl.dirkslot.encyclo.uk` `https://itunes.apple.com/de/app/encyclopedia-en/id964502197?mt=8` jeweils zuletzt aufgerufen am 12.08.2016 um 19:01

[35] `https://play.google.com/store/apps/details?id=org.wikipedia` `https://itunes.apple.com/us/app/wikipedia/id324715238?mt=8`
jeweils zuletzt aufgerufen am 12.08.2016 um 19:10

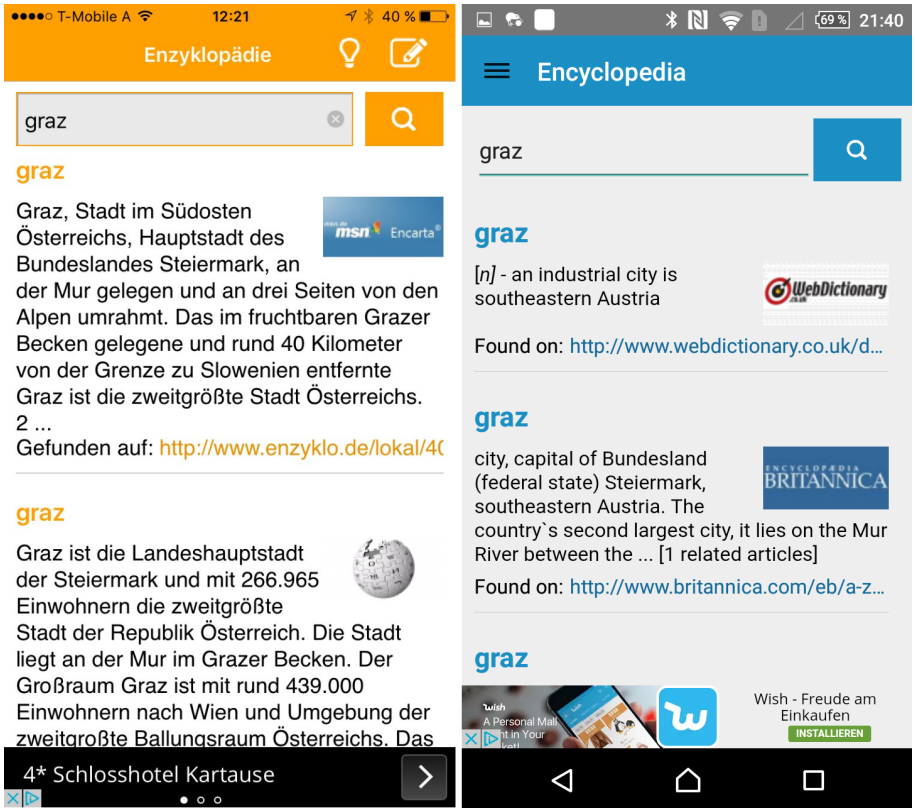

Abbildung 19: Enzyklopädie / Encyclopedia für iOS / Android

da alle Inhalte von der gleichnamigen Webseite kommen, die Darstellung jedoch in einer eigens für die mobile Applikation entwickelten Form umgesetzt ist. Wie auf Abbildung 20 zu sehen ist, werden alle bekannten Funktionen und Möglichkeiten der Wikipedia angeboten. Unter der Rubrik „Entdecken“ bekommen Benutzerinnen und Benutzer die Hauptseite der Wikipedia präsentiert. Innerhalb dieses Kontextes befinden sich auch die Featuers des zufälligen Artikels sowie der Artikel in der Nähe. Wenn allerdings gerade ein Artikel gelesen wird, muss zuerst zurück zur „Entdecken“ Seite navigiert werden. Auf dieser Seite angekommen, müssen Benutzerinnen und Benutzer zum gewünschten Feature scrollen um dieses verwenden zu können. Ein schneller „one-click“ Zugriff ist deshalb nicht möglich. Neben den bereits erwähnten Möglichkeiten wird natürlich auch eine Suche angeboten, bei der zielgerichtet nach Informationen gesucht werden kann.

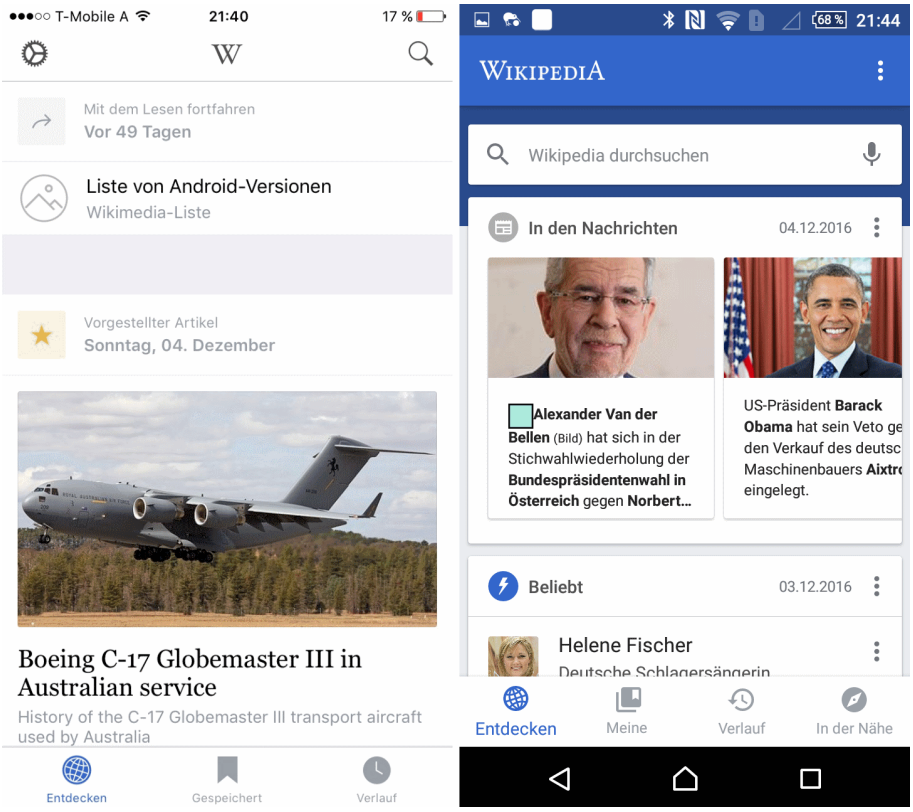

Abbildung 20: Die offizielle Wikipedia Applikation

17. **Brockhaus**[36]

    Brockhaus wirkt aufgrund der Beschreibungen, welche in den jeweiligen Stores hinterlegt sind, vielversprechend. Leider konnte diese Applikation weder für iOS noch für Android getestet werden, da nach dem Start verlangt wird einen Zugang einzurichten. Bei diesem Zugang handelt es sich im Grunde um einen VPN-Tunnel, der zu einer bestimmten Wissenseinrichtung aufgebaut werden muss. Es wird zwar eine Vielzahl an auszuwählenden Institutionen angeboten, wie beispielsweise Universitäten und Hochschulen, jedoch hat nicht jeder/ jede Benutzer/ Benutzerin einen Zugang zu solch einer Einrichtung. Weil sich diese Arbeit mit frei zugänglichem Wissen beschäftigt, wird diese Limitierung als Grund gesehen, dass diese Applikation nicht weiter betrachtet wird.

[36] `https://play.google.com/store/apps/details?id=de.brockhaus.wissen` `https://itunes.apple.com/de/app/brockhaus/id1126027419?mt=8` jeweils zuletzt aufgerufen am 19.08.2016 um 08:06

2.3.3.4 Zusammenfassung

Die Evaluierung der oben angeführten Applikationen zeigt, dass es verschiedene Wege und Techniken gibt, mit deren Hilfe Information Benutzerinnen und Benutzer präsentiert wird. Die Mehrheit der Applikationen (1-12, 16) verwendet als Informationsquelle Wikipedia. Für die Verifizierung dieser Behauptung wurden unter anderem einige Artikel als Stichproben gewählt. Diese wurden in der jeweiligen Applikation sowie auf Wikipedias Webseite miteinander verglichen. Einerseits wurde der Inhalt direkt verglichen und andererseits die Anordnung und Strukturierung der Artikel. Neben dem Abgleichen der Stichproben wurden für die Untermauerung der Behauptung zum Teil Hinweise in den Applikationen selbst oder in deren Beschreibung gefunden. Ein Beispiel für einen Hinweis ist in Applikation Nummer 8 zu finden. Hier wird der Begriff „Wikipedia“ als Platzhalter für die Suchleiste verwendet. Ein weiteres und wohl das offensichtlichste Indiz dafür, dass die Informationen von Wikipedia stammen, ist jedoch die Tatsache, dass Wikipedias Webseite schlicht in der Applikation geladen wird. Natürlich gibt es auch Applikationen, die ihre Informationen nicht nur von der Wikipedia beziehen. Nummer 4, 10, 14 und 15 bieten zusätzlich die Möglichkeit ihre Informationen von anderen Quellen zu beziehen.
Weitere Unterschiede sind auch in der Art und Weise, wie Information Benutzerinnen und Benutzer nahegebracht wird, zu finden. Hier kann vor allem zwischen den oben beschriebenen Wikipedia-Container und Wikipedia-Reader unterschieden werden. Applikation 1-3, 5-7 sowie 12 fallen unter die Kategorie des Wikipedia-Containers. Als Wikipedia-Reader können die Nummern 4, 8-11, 14 und 16 eingestuft werden. Dass die genannten Reader und Container überhaupt die Informationen verwenden dürfen, liegt an der Lizenzierung der Inhalte von Wikipedia. Diese sind unter der CC-by-sa-3.0 Lizenz zur Verfügung gestellt. Die Lizenz besagt unter anderem, dass die Inhalte vervielfältigt, verbreitet und öffentlich zugänglich gemacht werden dürfen. Die Bedingungen dafür sind zum einen Namensnennung und zum anderen, dass die Weitergabe unter gleichen Bedingungen erfolgt.[37]
Neben dem Ursprung und der Präsentation der Informationen kann auch dahingehend kategorisiert werden, wie und auf welchem Wege die einzelnen Applikationen ihre Informationen erhalten. Einerseits gibt es jene, die, damit sie vollständig funk-

[37] `https://de.wikipedia.org/wiki/Wikipedia:Lizenzbestimmungen_Creative_Commons_Attribution-ShareAlike_3.0_Unported/DEED` zuletzt aufgerufen am 18.11.2016 um 13:48

tionieren, eine aktive Verbindung zum Internet benötigen (2-10, 12, 13, 15-17), da sie den kompletten Inhalt je nach Bedarf laden. Andererseits gibt es auch jene, die nach Erstinbetriebnahme offline verwendbar sind (1, 11, 14). Letztere bieten also die Möglichkeit auch ohne oder mit einer schlechten Verbindung Information ohne Umwege aufzubereiten und zu präsentieren. Der Nachteil hier ist allerdings, dass bei Erstinbetriebnahme meistens eine große Menge an Daten herunterzuladen ist. Außerdem sind leider nicht alle offline Lösungen gratis. Lediglich Applikation Nummer 14 bietet mehrere Möglichkeiten Informationen aus verschiedenen Quellen in unterschiedlichen Größen herunterzuladen. Nummer 11 unterstützt zumindest gratis Pakete mit bis zu 180 Megabyte ohne weitere Kosten zu verursachen. Um Nummer 1 im offline Modus zu verwenden, kommen Benutzerinnen und Benutzer um zusätzliche Kosten nicht herum[38].

An dieser Stelle ist auch noch zu erwähnen, dass Wikipedia-Container für Benutzerinnen und Benutzer mit keinem technischen Hintergrund teilweise zu vielversprechend wirken können. Verantwortlich dafür ist, dass gewisse Container zwar die mobile Webseite der Wikipedia anzeigen, selbst aber nicht alle Funktionen von dieser unterstützen. Genauer gesagt, ist hier das Feature „Artikel in der Nähe“ gemeint. Diese Art der Suche benötigt gewisse Berechtigungen bzw. die Erlaubnis der Benutzerinnen und Benutzer um auf die aktuelle Position zugreifen zu dürfen. Wenn eine Container Applikation diese Funktionalität von sich aus anbietet, müssen dementsprechend bei der Entwicklung diese Bedingungen berücksichtigt werden. Um generell auf den aktuellen Standort des/ der Benutzers/ Benutzerin zugreifen zu dürfen, müssen bei Applikationen auf Android Basis bestimmte Elemente[39] mit den richtigen Attributen in das „App Manifest“[40] eingetragen werden. Um den aktuellen Ort bestimmen zu dürfen muss, genauer gesagt, das <`uses-permission`> Element mit dem passenden „Permission“[41] Attribut aufgenommen werden. Alle auf diesem Wege angeforderten Berechtigungen werden direkt bei der Installation angezeigt und

[38]Die hier beschriebenen Beobachtungen wurden tw. bereits zur Veröffentlichung von Neuhold et al. (2017) eingereicht

[39]`https://www.tutorialspoint.com/de/xml/xml_elements.htm` zuletzt aufgerufen am 11.12.2016 um 17:10

[40]`https://developer.android.com/guide/topics/manifest/manifest-intro.html` zuletzt aufgerufen am 11.12.2016 um 17:05

[41]`https://developer.android.com/reference/android/Manifest.permission.html` zuletzt aufgerufen am 11.12.2016 um 17:18

von den Benutzerinnen und Benutzer eingefordert, sofern die Applikation mit einer `targetSdkversion`[42] nicht größer als 22 ausgeliefert wurde. Mit API Version 23 hat Google das Berechtigungssystem von Android überarbeitet. Dieses besagt, dass nun Berechtigungen zur Laufzeit abgefragt werden müssen. Diese Veränderung soll den Installationsprozess von Applikationen schlanker gestalten. Benutzerinnen und Benutzer müssen dementsprechend nicht gleich bei der Installation alle Berechtigungen freigeben. Ebenfalls soll dadurch mehr Kontrolle über die Funktionalitäten der Applikation geboten werden. Benutzerinnen und Benutzer haben so unter anderem auch die Möglichkeit zu sehen, wann genau die geforderte Berechtigung benötigt wird, und können diese jederzeit widerrufen[43].
Zusätzlich zur eigentlichen Berechtigung müssen Container auf Android Basis eine ordnungsgemäße Kommunikation zwischen der WebView, welche die Berechtigung benötigt, und dem Container selbst gewährleisten. Um diese sicherzustellen muss zu Beginn für die WebView ein WebChromeClient[44] gesetzt werden. Dieser muss in weiterer Folge die Funktion `onGeoLocationPermissionsShowPrompt (String origin, GeolocationPermissions.Callback callback)`[45] überschreiben und korrekt implementieren. Im Grunde handelt es sich hier um eine Schnittstelle zwischen der in der WebView geladenen Webseite und dem Container. Versucht eine Webseite den aktuellen Ort zu bestimmen wird die zuvor genannte Funktion aufgerufen und es obliegt im weiteren Verlauf dem/ der verantwortlichen Entwickler/ Entwicklerin, dass an dieser Stelle der Callback mit den verfügbaren Rechten ausgeführt wird.

Das Verhalten auf iOS ist ebenfalls im gleichen Kontext betrachtet worden. Hier ist die sogenannte „Info.plist“[46] Datei als iOS-Äquivalent zum App-Manifest auf Android Seite zu betrachten. In diesem werden gewisse Basiskonfigurationen der Applikation festgelegt. Die Info.plist Datei besteht aus XML-Elementen, ähnlich

[42] `https://developer.android.com/guide/topics/manifest/uses-sdk-element.html` zuletzt aufgerufen am 11.12.2016 um 17:42

[43] `https://developer.android.com/training/permissions/requesting.html` zuletzt aufgerufen am 11.12.2016 um 17:51

[44] `https://developer.android.com/reference/android/webkit/WebChromeClient.html` zuletzt aufgerufen am 11.12.2016 um 18:00

[45] `https://developer.android.com/reference/android/webkit/WebChromeClient.html#onGeolocationPermissionsShowPrompt(java.lang.String,android.webkit.GeolocationPermissions.Callback)` zuletzt aufgerufen am 11.12.2016 um 18:08

[46] `https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Introduction/Introduction.html` zuletzt aufgerufen am 11.12.2016 um 19:47

wie bei Android das App Manifest, mit dem Unterschied, dass hier die Werte mit Hilfe von „Key-Value“ Paaren realisiert sind. Um nun auf die aktuelle Position des/ der Benutzers/ Benutzerin zugreifen zu dürfen muss in der Info.plist dem Schlüssel „NSLocationWhenInUseUsageDescription“[47] ein passender Wert zugewiesen werden. Passend bedeutet in diesem Kontext, dass dieser grundsätzlich mit einer sinnvollen Erklärung belegt sein sollte. An dieser Stelle ist bewusst der Konjunktiv verwendet worden, da diese Erklärung, unabhängig vom Inhalt, Benutzerinnen und Benutzer angezeigt wird, sobald die Applikation auf den Standort zugreifen will. Das Betriebssystem kann aber nicht entscheiden ob der anzuzeigende Wert grundsätzlich inhaltlich treffend ist und zum aktuellen Kontext passt. Die Verantwortung eine sinnvolle Beschreibung zu wählen obliegt dem/ der verantwortlichen Entwickler/ Entwicklerin.
Wird nun von dem Fall ausgegangen, dass eine Webseite, die in einer UIWebView bzw. WKWebKit geladen wird, auf die aktuelle Position des/ der Benutzers/ Benutzerin zugreifen will, so wird anders als bei Android die Kommunikation zwischen der verwendeten Technologie und dem Container automatisch abgewickelt. Es wird lediglich auf den Wert, der in dem Schlüssel „NSLocationWhenInUseUsageDescription“ hinterlegt ist, zurückgegriffen und angezeigt. Wenn aktive Benutzerinnen und Benutzer diesem Dialog zustimmten, dann darf die Webseite die Position verwenden um beispielsweise Artikel in der Nähe anzuzeigen. Ist dieser Schlüssel schlichtweg nicht gesetzt oder kein Wert vorhanden, dann darf die geladene Webseite auch nicht den aktuellen Standpunkt des/ der Benutzers/ Benutzerin verwenden.

Die obig beschriebenen Verhaltensweisen wurden dadurch bestätigt, dass zu Testzwecken jeweils ein Container für Android sowie für iOS entwickelt worden ist, Container, welche lediglich Webseiten eingebunden haben, mit dem Ziel auf die aktuelle Position des Gerätes zuzugreifen.

[47] `https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple_ref/doc/uid/TP40009251-SW26` zuletzt aufgerufen am 11.12.2016 um 20:20

3 Umsetzung

Für diese Masterarbeit wurde festgelegt, dass das Prinzip des Prototyping umgesetzt werden soll. Dies bedeutet, dass ein Prototyp für Android und iOS entwickelt worden ist. Folgendes Kapitel behandelt zunächst den Begriff des Prototypings sowie die Auswertung der zuvor durchgeführten Evaluierung. Anschließend wird die Umsetzung des resultierenden Prototypen beschrieben.

3.1 Prototyping

Als Prototyping ist der Prozess zu bezeichnen, der alle Tätigkeiten umfasst, die notwendig sind um einen Prototypen zu entwickeln. Die Erstellung eines Software-Prototypen soll im Gegensatz zu anderen Prototypen schnell und kostengünstig möglich sein. Des Weiteren sollen Software-Prototypen ausführbare Systeme sein, die alle wesentlichen Eigenschaften des Zielsystems erfüllen. Dadurch soll ermöglicht werden, kritische Aspekte der resultierenden Software zu identifizieren. Eventuell gedachte Schwierigkeiten sollen damit getestet und das Verhalten in einer realistischen Umgebung überprüft werden. Entpuppt sich eine angedachte Schwierigkeit tatsächlich als Problem, so ist dieses bereits in einem frühen Stadium bekannt. Dadurch kann darauf reagiert werden und die Anforderungen dementsprechend angepasst und abgeändert werden (vgl. Kuhrmann, 2012) (vgl. Steinbauer and Thomas, 2003, Seite 7) (vgl. Pomberger et al., Seite 2).
Der Prototyp orientierte Software-Life-Cycle unterscheidet sich demnach von den klassischen Entwicklungsstrategien nur darin, dass es nicht bloß als lineares Phasenmodell sondern vielmehr als iteratives Modell zu sehen ist (vgl. Steinbauer and Thomas, 2003, Seite 4-5).
Es ist auch nicht konkurrierend sondern vielmehr als ergänzend zu den klassischen Strategien zu betrachten, sodass ein modifiziertes Life-Cycle-Modell entstehen kann, bei dem der strenge sequentielle Ablauf bei bestimmten Stellen durch das angesprochene iterative Verhalten ersetzt bzw. ergänzt werden kann (vgl. Pomberger et al., Seite 3).

3.2 Auswertung der Evaluierung

Aufgrund der zuvor durchgeführten Evaluierung der im Kapitel „Stand der Technik“ beschriebenen Applikationen haben sich einige nennenswerte Features ergeben. Um nicht bloß einen Container für ein Nachschlagewerk zu entwickeln wurden die beobachteten Funktionalitäten in einer Expertenrunde besprochen, diskutiert und erweitert. Dabei wurde stets der Gedanke mobile Endgeräte und ihre technischen Möglichkeiten auszureizen, bedacht und umgesetzt. Ergebnis der Evaluierung und der Diskussionsrunden sind folgende aufgelistete Key-Features. Für den weiteren Verlauf der Entwicklung des Prototypen werden aus den Beschreibungen anschließend User-Stories gebildet. Für die Erstellung dieser wurde folgende Form verwendet:

```
Als <Rolle> möchte ich <Ziel/Wunsch>, um <Nutzen>(optional)
```

1. **Zufälliger Artikel**
   Bei einem zufälligen Artikel geht es darum die Möglichkeit zu bieten einen zufälligen Artikel aus der gesamten verfügbaren Datenbank zu erhalten. Dies ist vor allem für jene Benutzerinnen und Benutzer relevant, die einerseits nach neuem Wissen streben, aber andererseits gerade nicht wissen, was für Informationen sie abrufen sollen. Mit dieser Möglichkeit wird Benutzerinnen und Benutzern die Chance geboten ihren Wissensdurst ohne viel Aufwand oder Mühen zu stillen. Dabei soll Benutzerinnen und Benutzern möglichst wenig Interaktion abverlangt werden. Aus den Expertenrunden ging auch hervor, dass Benutzerinnen und Benutzer bei der Benutzung dieses Features zusätzlich die Einstellungen dahingehend modifizieren können, dass eine bestimmte Kategorie ausgewählt werden kann. Zufällige Artikel sind dann zumindest dieser Kategorie zugehörig. Diese Einstellungsmöglichkeit reduziert somit den absoluten Zufallsfaktor und bietet Benutzerinnen und Benutzer einen Weg die Funktion an deren Verhalten und Wünsche anzupassen.

   User-Story
   Eine anwendende Person möchte gerne etwas Unbekanntes lesen um damit neues Wissen zu generieren, hat aber keine Idee, wonach sie suchen soll.

2. **Artikel des Monats**

   Ebenfalls wurde beschlossen, dass als weiteres wichtiges Feature der sogenannte „Artikel des Monats“ in den resultierenden Prototypen aufzunehmen ist. Allerdings wurde festgehalten, dass dieser nicht limitiert auf einen einzelnen speziellen Artikel ist. Vielmehr wird die Auswahl an Hand einer Liste getroffen, welche alle neuen Artikel des aktuellen Monats enthaltet. Die Auswahl des anzuzeigenden Artikels wird dabei per Zufall entschieden. Für den/ die Benutzer/ Benutzerin selbst, agiert diese Logik ähnlich wie die beim zufälligen Artikel, jedoch wird hier der Zufallsfaktor drastisch reduziert. Somit wird Benutzerinnen und Benutzern aktuell relevantes und neues Wissen vermittelt.

   User-Story

   Benutzerinnen und Benutzer möchten gerne etwas Neues, was aktuell in diesem Monat von Bedeutung ist, in Erfahrung bringen um zeitnahe noch unbekannte Information zu erhalten.

3. **Artikel in der Nähe**

   Das mit Abstand wichtigste und am meisten durch die Expertenrunde diskutierte Feature für den resultierenden Prototypen ist die so genannte „Artikel in der Nähe“ Funktionalität. Bei der Planung dieses Features wurde vor allem darauf Wert gelegt, dass es die Möglichkeiten eines mobilen Gerätes ausreizt. Zielgruppe sollen hier jene Personen sein, die wissbegierig sind und ähnlich wie bei den anderen Features sich nicht entscheiden können, was genau sie lesen sollen, aber dennoch informiert werden wollen über ihre unmittelbare Umgebung. Ein wichtiger Unterschied hierbei ist, dass Benutzerinnen und Benutzer nicht irgendeine zufällige Information erhalten sondern direkt das Ergebnis, welches ihnen präsentiert wird, in einer gewissen Art und Weise beeinflussen. Dies geschieht dadurch, dass für das Bereitstellen der notwendigen Artikeln die aktuelle Position des/ der Benutzers/ Benutzerin verwendet wird. Genauer gesagt werden anhand des detaillierten Standpunktes des mobilen Gerätes etwaige Artikel in der Nähe berechnet.

   Die Wichtigkeit dieser Funktionalität wird durch die Tatsache betont, dass es sogar eigene Applikationen auf dem Markt gibt, die genau nur diese Funk-

tion umsetzen. Beispiele für solche wären zum einen WikiExplorer für iOS und zum anderen NearbyWiki für Android. Jedoch wurde bei diesen und auch bei anderen immer nur ein aktiver Ansatz beobachtet. Zusätzlich dazu wurde beschlossen, dass für den resultierenden Prototyp auch ein passiver Modus umgesetzt werden soll. Dieser Modus soll es dem/ der Benutzer/ Benutzerin ermöglichen, dass er/ sie über Artikel in der Nähe informiert wird, wenn er/ sie in Bewegung ist. Die Kommunikation soll dabei mit Hilfe des Benachrichtigungssystems der jeweiligen Betriebssysteme umgesetzt werden. Um Benutzerinnen und Benutzer mehr Kontrolle über die Vorgehensweise der Applikation zu geben wurde beschlossen, gewisse Einstellungsmöglichkeiten für das Verhalten der Lokalisierung sowie die Häufigkeit von Benachrichtigungen bereitzustellen. Dadurch kann jeder/ jede Benutzer/ Benutzerin dieses Feature seinen/ ihren Bedürfnisse anpassen.

User-Stories

- ***Artikel in der Nähe – Aktiv***
  Benutzerinnen und Benutzer befindet sich an irgendeinem Ort und wollen wissen, was für relevante Artikel es in ihrer Nähe gibt.
- ***Artikel in der Nähe – Passiv***
  Benutzerinnen und Benutzer sind in Bewegung und wollen laufend über Artikel in ihrer Nähe benachrichtigt werden.

4. **Textbasierte Suche**
   Neben den bereits genannten Möglichkeiten wurde ebenfalls beschlossen, dass Benutzerinnen und Benutzer auch explizit nach Informationen suchen können sollen. Dies soll jene Bedürfnisse abdecken, bei denen Benutzerinnen und Benutzer über ein bestimmtes Thema oder Stichwörter recherchieren wollen, sodass auch ein zielorientierter Weg für Informationsbeschaffung zur Verfügung gestellt wird.

   User-Story
   Benutzerinnen und Benutzer würden gerne zu einem für sie relevantes Thema oder Stichwort zusätzliche Information und Artikel erhalten.

5. **Lizenz Information über angezeigte Artikel**
   Das Thema Lizenz bzgl. gerade gelesenen Artikel spielt in der heutigen Zeit eine wichtige Rolle. Dies wurde bei Gesprächen in einer Expertenrunde so festgestellt. Dementsprechend ist beschlossen worden, dass es mehr oder weniger eine passive Lösung geben soll bei der Benutzerinnen und Benutzer gleich und ohne viel Umwege erkennen können, unter welchen Bedingungen die aktuelle Information lizenziert ist, sodass gleich bekannt ist, wie und ob das gerade Gelesene weiter zu verwenden ist.

   User-Story
   Die Benutzerinnen und Benutzer würden gerne wissen unter welcher Lizenz der gerade aktive Artikel steht, damit sie über die Verwendung des Inhaltes Bescheid wissen.

6. **Datenpersistenz**
   Zusätzlich zu den oben genannten Möglichkeiten Informationen zu erhalten und abzurufen wurden auch Diskussionen bzgl. Datenpersistenz geführt. Eine wesentliche Entscheidung, die an Hand derer getroffen wurde, ist, dass für die Verwendung des Prototypen eine aktive Internetverbindung bestehen muss. Ohne diese kann der Prototyp nicht in vollem Ausmaß benutzt werden. Dies ist vor allem aus zwei Gründen beschlossen worden. Zum einen soll vermieden werden, dass Benutzerinnen und Benutzer zusätzliche Wartezeit aufbringen müssen, nachdem sie die Applikation installiert haben. Denn würde es sich um eine offline Lösung handeln, dann müssten Benutzerinnen und Benutzer zumindest nach Erstinstallation etwaige Datenpakte mit Inhalten herunterladen. Zum anderen soll die stetig wachsende und aktualisierte Wissensbasis zur Verfügung gestellt werden. Um also zu vermeiden, dass stetig Datenupdates geladen und installiert werden müssen, sollen Inhalte nur nach Bedarf angefordert werden. Dies soll auch dazu führen, dass der benötigte Datentransfer minimiert wird. Neben der eigentlichen Informationsgewinnung soll es in diesem Zusammenhang für Benutzerinnen und Benutzer auch möglich sein Lesezeichen für Artikel zu setzen, sodass es ihnen möglich ist, gerade begonnene

Artikel später fertig zu lesen.

User-Stories

- ***Informationsbeschaffung***
  Benutzerinnen und Benutzer möchten nach Erstinstallation gleich und ohne Umwege den vollen Funktionsumfang der Applikation genießen.
- ***Lesezeichen***
  Benutzerinnen und Benutzer möchten einen Artikel den sie gerade lesen, als Lesezeichen speichern, damit sie diesen später schnell griffbereit haben.

7. **Home- / Navigation**
   Dieses Feature wirkt auf den ersten Blick eher trivial sollte aber nicht unterschätzt werden. Navigation alleinstehend beschreibt die Funktionsweise ähnlich wie die eines Verlaufs. Damit soll ermöglicht werden, dass Benutzerinnen und Benutzer im Verlauf ihrer besuchten bzw. gelesenen Artikeln vor- und zurückspringen können. Dies wird an dieser Stelle hier explizit erwähnt, da diese simple jedoch sehr angenehme Funktion nicht bei allen evaluierten Applikationen geboten wurde.
   Ähnliches wurde bei der Home-Navigation beobachtet. Diese wurde bei manchen Applikationen sehr umständlich verwirklicht und teilweise gar nicht geboten. Bei der Umsetzung einer „one-click“ Lösung für das erneute Laden des Startbildschirms ist an jene Benutzerinnen und Benutzer gedacht worden, die die Wissensbasis und all ihre Kategorien durch exploratives Navigieren erkunden wollen.

   User-Stories

   - ***Navigation***
     Benutzerinnen und Benutzer klicken auf einen verlinkten Artikel und würden gerne zurück navigieren um den zuvor aktiven Artikel weiterzulesen.

- ***Home***
  Benutzerinnen und Benutzer möchten gerne wieder zurück zur Austria-Forum Startseite um neue Kategorien durch Navigieren zu erforschen.

3.3 Technische Umsetzung

3.3.1 User-Interface

Der im Rahmen dieser Arbeit entstandene Prototyp einer mobilen Applikation für große Nachschlagewerke stellt unter anderem folgenden Leitsatz an sich selbst: Möglichst einfach und schnell neues Wissen zu generieren und Benutzerinnen und Benutzer zur Verfügung zu stellen. Ausgehend von dem genannten Leitsatz sowie folgender Aussage von Neubauer (2016)

> *[...]Das User-Interface (UI) bestimmt die Klarheit. Wenn man so will ist UI wie ein Witz: Muss man es erklären, ist es nicht gut. Eine App muss also möglichst intuitiv nutzbar sein, Design und Inhalte müssen sich selbst beschreiben. Je besser das gelingt, desto schneller können neue Nutzer produktiv damit umgehen.[...]*

wurde versucht das User-Interface möglichst einfach und intuitiv zu gestalten um eine positive User-Experience zu schaffen. Denn ein Viertel der Benutzerinnen und Benutzer benutzten eine heruntergeladene mobile Applikation nur genau ein einziges Mal. Dies ist zurückzuführen auf eine schlechte User-Experience, welche sich dann ergibt wenn Benutzerinnen und Benutzer anfangs mit dem Funktionsumfang überfordert sind und zu viel Zeit benötigen um sich zu orientieren (vgl. Weddehage, 2016).
Weddehage weist auch darauf hin, dass der „Eincheck Prozess“ sehr wichtig ist. Er betont, dass in diesem Benutzerinnen und Benutzer den Funktionsumfang der Applikation genauer kennen lernen sollen und gleichzeitig den Nutzen, der sich dadurch ergibt, wahrnehmen sollen (vgl. Weddehage, 2016).
Dies wurde umgesetzt, indem beim Start des Prototypen direkt die Startseite des Austria-Forums geladen wird. Auf Abbildung 21 ist der Startbildschirm der mobilen Applikation zu sehen. An dieser Stelle ist eine grobe Aufteilung der gesamten Wissensbasis in ein paar übergeordnete Kategorien zu erkennen. Damit soll dem/

der Benutzer/ Benutzerin zum einen die Masse an Information vor Augen geführt werden und zum anderen werden dadurch gewisse Interessen der Benutzerinnen und Benutzer angesprochen.

Ebenfalls wurde entschieden, dass möglichst wenig Aufwand von Benutzerinnen

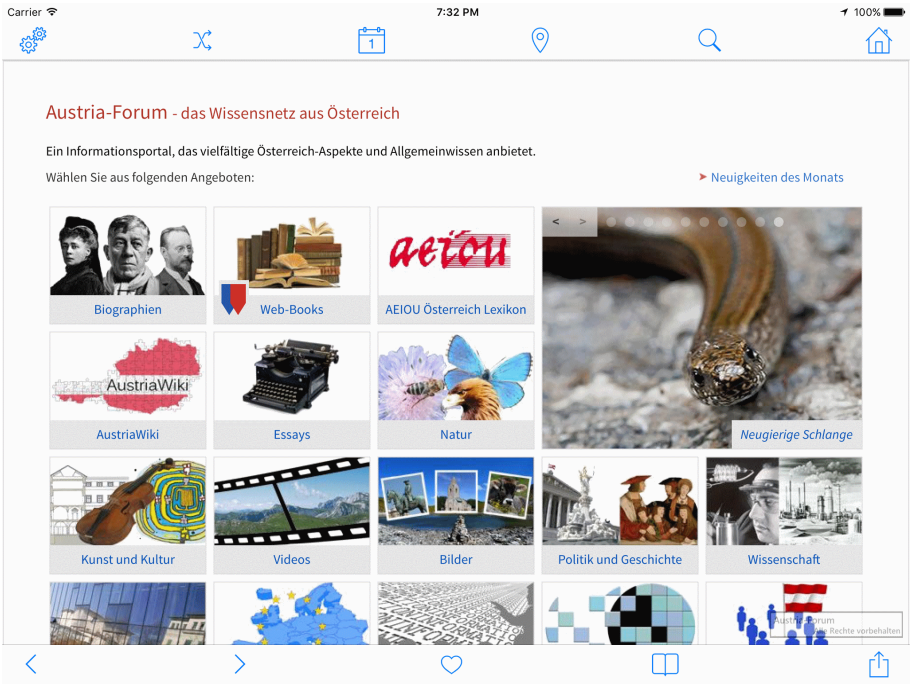

Abbildung 21: Startscreen des Prototypen, iOS

und Benutzer abverlangt werden soll um etwaige Funktionen der Applikation durchzuführen. Dieser Ansatz beruht zum einen auf der Tatsache, dass Benutzerinnen und Benutzer den Weg des geringsten Widerstandes wählen. So beschreiben es zumindest Tétard and Collan (2009) in deren Artikel über die „Lazy User Theory“.
Zum anderen ist zu erwähnen, dass heutzutage alles sehr schnell gehen muss. Das Motto der heutigen Gesellschaft könnte mit „Faster! Faster! Faster!“ beschrieben werden (vgl. Hammelehle, 2013).
Weitere Studien haben auch gezeigt, dass bereits eine Sekunde Verzögerung beim Laden einer Webseite die Zufriedenheit der Benutzerinnen und Benutzer um 16% senkt. (vgl. Weatherhead, 2014)
Deswegen wurde entschieden, dass das User-Interface dementsprechend einfach gehalten wird, sodass eine etwaige Benützung möglichst wenig Aufwand erfordert.

> *[...]Visueller Content kann das menschliche Gehirn besser aufnehmen und deutlich schneller verarbeiten als Information in Textform[...]*(vgl. Weddehage, 2016)

Ausgehend davon ist beschlossen worden, dass für die Applikation aussagekräftige Grafiken zu wählen sind, sodass auf eine textuelle Beschreibung der Funktionen verzichtet werden kann. Zusätzlich, um dem Motto der heutigen Gesellschaft zu entsprechen, ist ebenfalls entschieden worden, dass die Grafiken in Toolbars eingebunden werden, damit diese zentral und vor allem schnell erreichbar sind. Die Platzierung der Toolbars kann in Abbildung 21 betrachtet werden. Für eine detailliertere Ansicht wird an dieser Stelle auf Abbildung 22 verwiesen.
Eine weitere Regel für ein gutes User-Interface besagt, dass Benutzerinnen und Be-

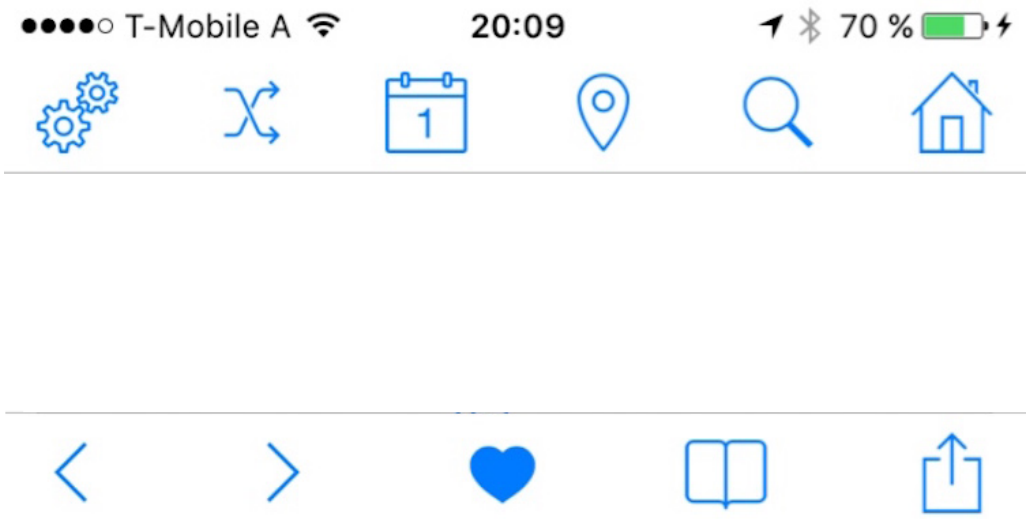

Abbildung 22: Detailierte Ansicht der erstellten Toolbars und der verwendeten Grafiken

nutzer immer auf dem Laufenden gehalten werden sollen über alles, was gerade im Programm vor sich geht. Auf keinen Fall darf es so weit kommen, dass die Anwender/ Anwenderinnen überlegen und raten was gerade passiert (vgl. Babich, 2016).
Neben einem Ladebalken, der das Laden der angeforderten Austria-Forum Inhalte signalisiert, gibt es zusätzlich bei jeder Kommunikation zwischen Applikation und API weitere Statusmeldungen. Ist der Server vorübergehend nicht erreichbar oder kommt es generell zu Verbindungsproblemen wird eine dementsprechende Meldung angezeigt. Damit wird bezweckt, dass auch bei einer schlechten Internetverbindung

die Benutzerinnen und Benutzer stets auf dem Laufenden gehalten werden, was gerade in der Applikation vor sich geht.

3.3.1.1 Technische Details - UI

iOS

Die Architektur des User-Interface bzw. des erstellten Storyboards[48] ist in Abbildung 23 zu sehen. Zu erkennen ist, dass als initialer ViewController ein UINavigationController[49] eingesetzt worden ist. Als "root view controller" des Navigation Controllers wurde ein UIViewController[50] verwendet, welcher in weiterer Folge als DetailViewController bezeichnet wird, nicht zu verwechseln mit einem DetailViewController eines UISplitViewControllers. Beim DetailViewController des Prototypen handelt es sich nämlich um die Hauptanzeige der Applikation. In diesem Controller werden Artikel geladen und alle Details bzgl. Inhalt und Lizenz präsentiert. Ebenfalls zuständig ist dieser für die Anzeige der Toolbars und die damit verbundene Delegation der Verantwortungen an die einzelnen Programmteile. Ausgehend vom DetailViewController wird abhängig von der Aktion der Benutzerinnen und Benutzer ein neuer ViewController auf den Navigationsstack des NavigationController gelegt. Dies wurde mit Hilfe von Xcode's Interface-Builder realisiert, indem eine sogennante „Storyboard Segue"[51] zwischen dem gedrückten Icon aus der Toolbar und dem zuständigen ViewController hergestellt wurde. Eine Segue ist zuständig für den visuellen Übergang zwischen zwei ViewController. Nach erfolgreichem Übergang befindet sich der neue ViewController an oberster Stelle des Navigationsstacks. Die Navigation zum vorherigen ViewController erfolgt dadurch, dass der aktuelle ViewController vom Stack entfernt wird. Für die Visualisierung der Toolbars wurde eine eigene Klasse erstellt, welche von der UIToolbar-Klasse abgeleitet worden ist. Im Interface-Builder wurde diese als „Custom Class" für die jeweiligen UIToolbar Elemente gesetzt. Ausgehend von der Definition der ToolBar-Klasse wurden passende

[48] `https://developer.apple.com/library/content/documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html` zuletzt aufgerufen am 16.12.2016 um 13:15

[49] `https://developer.apple.com/reference/uikit/uinavigationcontroller` zuletzt aufgerufen am 16.12.2016 um 13:19

[50] `https://developer.apple.com/reference/uikit/uiviewcontroller` zuletzt aufgerufen am 16.12.2016 um 13:25

[51] `https://developer.apple.com/reference/uikit/uistoryboardsegue` zuletzt aufgerufen am 16.12.2016 um 14:09

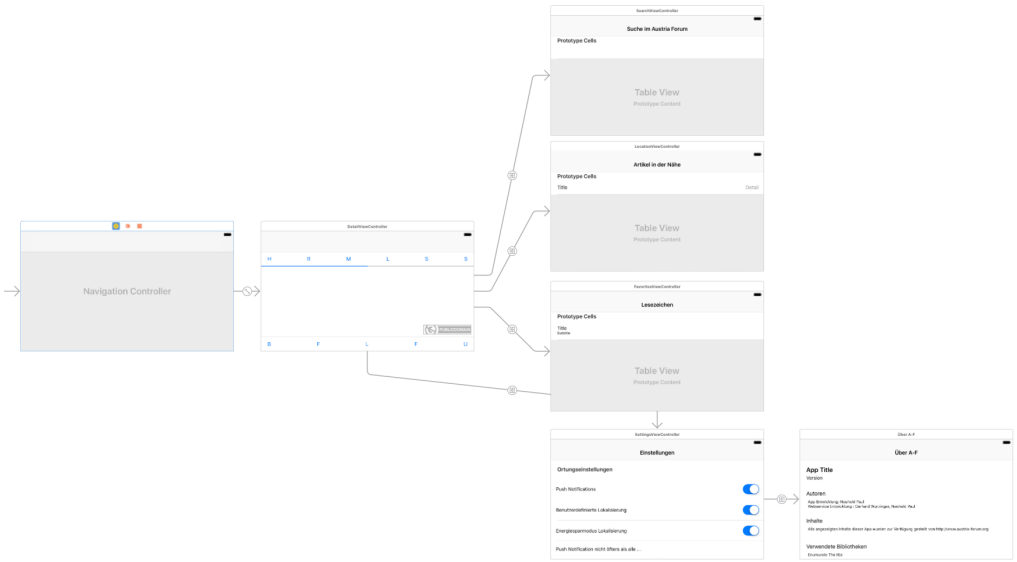

Abbildung 23: Storyboard Austria-Forum für iOS

Werte für die „Tags“ der UIBarButtonItems bestimmt und anhand dieser werden während der Generierung der View die passenden Icons geladen. Aufgrund der Tatsache, dass es keinen standardisierten Zugriff auf die UIImage-Klasse gibt, welche bei Erstellung des Objekts eine Größe erlaubt, die explizit angegeben werden kann, wurde beschlossen eine Extension dafür zu implementieren[52].
Denn für die Buttons der Toolbar werden Objekte des Typen UIImage benötigt. Zwar gibt es Möglichkeiten mit einem gewissen Skalierungsfaktor die Objekte zu erstellen oder die zu verwendenden Grafiken direkt in der passenden Größe bereitzustellen. Jedoch wurden diese Möglichkeiten ausgeschlossen, da hier eventuelle Anpassungsmöglichkeiten schwerer umzusetzen sind. Des Weiteren ist damit der Grundstein für eine dynamische Anpassung der Größe der Icons durch Benutzerinnen und Benutzer gelegt worden. Dies ist bei der aktuellen Version des Prototypen im Backlog aufgenommen, aber noch nicht umgesetzt worden. Die zuvor angesprochene Extension der UIImage-Klasse ist in Listing 1 zu sehen. Die zu sehende Funktion erwartet den Namen des Bildes und die gewünschte Größe. Ausgehend von den Parametern wird ein Bild als UIImage Typ erzeugt und in eine mit gegebener Größe erzeugten UIImageView platziert. Die so erzeugte UIImageView wird in weiterer Folge in einem grafischen Kontext gerendert. Mit der passenden Funktion ist

[52] `https://developer.apple.com/reference/uikit/uiimage` zuletzt aufgerufen am 17.12.2016 um 17:19

es anschließend möglich den aktuellen grafischen Kontext als UIImage zu erhalten. Dieses so generierte Objekt wird zum Abschluss an den Caller zurückgegeben und anschließend verwendet um das Bild der UIBarButtonItems zu setzen.

```
extension UIImage {
    class func renderedImageInGraphicContext(name: String, size: CGSize)
        -> UIImage?
    {
        let image = UIImage(named: name)
        let imageViewForRendering = UIImageView(frame: CGRectMake(0, 0,
            size.width, size.height))
        var renderedImage : UIImage?

        if let unwrapped_image = image
        {
            imageViewForRendering.image = unwrapped_image
            UIGraphicsBeginImageContextWithOptions(size, false, 0)
            if let currentGraphicContext = UIGraphicsGetCurrentContext()
            {
                imageViewForRendering.layer.renderInContext(
                    currentGraphicContext)
                renderedImage =
                    UIGraphicsGetImageFromCurrentImageContext()
            }
            UIGraphicsEndImageContext()
        }
        return renderedImage
    }
}
```

Listing 1: UIImage+Extension

Android

Die generelle Struktur und der Aufbau des User-Interface ist auf Android Seite ähnlich wie jenes des Prototypen für iOS. Die zu Grunde liegenden technischen Hilfsmittel umfassen klarerweise jene des Android SDK. Allerdings können die logischen Programmabläufe, welche zuständig für Visualisierung und Präsentation der einzelnen Elemente des User-Interfaces, sind auf einer abstrahierten Ebene mit jenen der iOS Version gleichgesetzt werden. Für das Anzeigen der einzelnen Artikel und der Toolbars sowie die Delegation der Aufgaben ist hier die MainActivity des

Programms verantwortlich. Diese ist gleichzeitig auch der Startpunkt der Applikation. Je nach Aktion der Benutzerinnen und Benutzer werden bestimmte Features ausgeführt oder eben auch weitere Activities gestartet und auf den sogenannten „Back Stack“[53] gelegt. Dank der internen Struktur des Android Frameworks und dem damit verbundenen Back Stack können für die Navigation Activities erstellt werden und anschließend, wenn die Arbeit vollendet ist, wieder mit einem Aufruf der Methode „`finish()`“ beendet werden. Die so beendete Activity wird vom Stack entfernt und die darunter liegende wird wieder angezeigt.
Die Toolbars werden durch ein „Linear Layout[54]“ mit horizontaler Ausrichtung und passenden Sub-Elementen bestimmt.

3.3.2 Informationsbeschaffung

Wie in letzterem Kapitel bereits beschrieben, wird heutzutage von Benutzerinnen und Benutzern erwartet, dass Applikationen möglichst schnell arbeiten. Es darf keine Verzögerung geben. Genau deswegen wurde beschlossen, dass bei der Informationsbeschaffung auf eine Onlinelösung gesetzt wird. Dadurch soll verhindert werden, dass Benutzerinnen und Benutzer nach Installation erst eine große Datenmenge herunterladen müssen, bevor sie mit der Applikation arbeiten können.
Sämtliche Inhalte, die in dem Prototypen verwendet und angezeigt werden, stammen vom Austria-Forum[55]. Das Austria-Forum selbst bezeichnet sich als eine Wissens- und Diskussionsplattform mit Schwerpunkt Österreich, welche von unabhängigen Wissenschaftlern/ Wissenschaftlerinnen und Publizisten/ Publizistinnen gestaltet und der Allgemeinheit zur Verfügung gestellt wird. Zusätzlich dazu ist das Austria-Forum eine Informationssammlung, die regional und zeitlich in die Tiefe geht. Zielgruppe des Austria-Forums sind alle, die an Information über Österreich bzw. Österreich-relevantem Wissen interessiert sind.[56]
Damit der Prototyp mit diesen Informationen arbeiten kann und auf diese zugreifen kann, musste für die Entwicklung zu Beginn die bestehende API des austria-forums.org dahingehend erweitert werden, dass sie den Ansprüchen der nativen

[53] `https://developer.android.com/guide/components/tasks-and-back-stack.html` zuletzt aufgerufen am 18.12.2016 um 12:52
[54] `https://developer.android.com/guide/topics/ui/layout/linear.html` zuletzt aufgerufen am 16.01.2017 um 21:33
[55] `http://austria-forum.org/` zuletzt aufgerufen am 14.08.2016 um 11:39
[56] `http://austria-forum.org/af/Infos_zum_AF/Grunds%C3%A4tze` zuletzt aufgerufen am 20.08.2016 um 15:42

mobilen Applikation gerecht wird. Bei der, dem Austria-Forum zugrundeliegende Technologie, handelt es sich um ein JSPWiki. JSPWiki ist eine freie Wiki-Software, die mit Hilfe der Standard JEE Komponenten geschrieben worden ist.[57]
Nach Gesprächen mit dem hauptverantwortlichen Programmierer des Austria-Forums wurde beschlossen, dass auf JSON-RPC bei der Weiterentwicklung der API gesetzt wird, da diese Möglichkeit vom JSPWiki angeboten wird.
RPC (Remote Procedure Call) bezeichnet ein Entwurfsmuster, mit welchem es möglich ist mit Client seitigem Code direkt Serverseitige Logik auszulösen. Ausgelöst wird diese beispielsweise durch eine so genannte JSON-RPC Anfrage. Diese beinhaltet ein JSON Objekt mit Informationen, welche Methode aufzurufen ist. Für die Antwort wird vom Server ebenfalls ein JSON Objekt zurückgesendet (vgl. Wahli et al., 2009, Seite 872-873).
JSON (JavaScript Object Notation) ist ein schlankes Datenaustausch Format, welches für Maschinen einfach zu parsen und für Menschen einfach zu lesen ist (vgl. Wahli et al., 2009, Seite 91).
JSON-RPC kann demnach als schlanker RPC bezeichnet werden (vgl. Hodge and and Tim Tripcony, 2012, Seite 351).
Aktuell gibt es zwei Spezifikationen des JSON-RPC, 1.0 sowie 2.0. Da in der Austria-Forum Architektur der 1.0 Standard verwendet wird, wird in folgendem auf diesen eingegangen. Für den erläuterten Aufbau der JSON-RPC Anfrage sowie Antwort wurde auf die offizielle JSON-RPC Spezifikation zurückgegriffen[58].

Eine JSON-RPC Anfrage setzt sich aus folgenden Teilen zusammen:

- `method`: Ein String, der den Namen der aufzurufenden Methode beinhaltet.
- `params`: Ein Array von Objekten, welche als Argumente der aufgerufenen Methode übergeben werden.
- `id`: Die id der Anfrage. Diese ist typenunabhängig und wird verwendet um vom Server kommende Antworten den passenden Anfragen zuzuordnen.

Die Antwort auf eine JSON-RPC Anfrage ist folgend definiert:

- `result`: Ein Objekt bestehend aus dem Rückgabewert der aufgerufenen Funktion. Im Fehlerfall muss dieser Wert `Null` sein.

[57] `https://jspwiki.apache.org/` zuletzt aufgerufen am 19.08.2016 um 13:22
[58] `http://json-rpc.org/wiki/specification` zuletzt aufgerufen am 20.08.2016 um 09:40

- `error`: Dieser Parameter beschreibt ein Fehlerobjekt, wenn es zu einem inkorrekten Aufrufen der Methode gekommen ist. Wurde kein Fehler ausgelöst, muss dieser Wert `Null` sein.
- `id`: Die id muss mit der in der Anfrage definierten `id` übereinstimmen um die Antwort der Anfrage zuordnen zu können.

3.3.2.1 Umsetzung API-Erweiterung des AF-Server

Wie oben schon erwähnt, unterstützt JSPWiki das Handling von JSON-RPC. Dafür stellt JSPWiki die Klasse JSONRPCManager[59] zur Verfügung. Mit Hilfe dieser werden alle Methoden einer Klasse, die das Interface RPCCallable[60] implementieren, für JSON-RPC Anfragen bereitgestellt. Zuvor muss die RPCCallable implementierende Klasse noch via `JSONRPCManger.registerGlobalObject(String id, RPCCallable object)`[61] registriert werden. Mit der SearchManager.JSONSearch Klasse[62] stellt die JSPWiki Software eine JSON-RPC API mit Zugriff auf die interne Suchmaschine zur Verfügung. Die Funktionalitäten, die der Prototyp hauptsächlich benötigt, beschäftigen sich in erster Linie mit dem Ausfindigmachen von Artikeln. Deshalb wurde beschlossen, dass die benötigten Erweiterungen direkt in die SearchManager Klasse aufzunehmen sind.

Ein typische Client-Server-Kommunikation soll in folgendem Beispiel näher beschrieben werden:

1. Zuerst schickt der Client eine Anfrage wie in Listing 2 via POST-Methode an den Server.

Die hier angegebenen Parameter entsprechen der JSON-RPC Spezifikation 1.0. Speziell die Inhalte der Parameter „method“ und „params“ sind hier von Bedeutung.

[59] `http://jspwiki.apache.org/apidocs/2.10.1/org/apache/wiki/rpc/json/JSONRPCManager.html` zuletzt aufgerufen am 20.08.2016 um 11:00

[60] `http://jspwiki.apache.org/apidocs/2.10.1/org/apache/wiki/rpc/RPCCallable.html` zuletzt aufgerufen am 20.08.2016 um 11:00

[61] `http://jspwiki.apache.org/apidocs/2.10.1/org/apache/wiki/rpc/json/JSONRPCManager.html#registerGlobalObject(java.lang.String,org.apache.wiki.rpc.RPCCallable` zuletzt aufgerufen am 20.08.2016 um 11:02)

[62] `http://jspwiki.apache.org/apidocs/2.10.1/org/apache/wiki/search/SearchManager.JSONSearch.html` zuletzt aufgerufen am 20.08.2016 um 11:13

Der dem Punkt vorangestellte Wert „search“ im Parameter „method“ beschreibt die Bezeichnung, unter welchem das RPCCallable implementierende Objekt zuvor mit Hilfe von `JSONRPCManger.registerGlobalObject(String id, RPCCallable object)` registriert wurde.

```
{
    "id":10004,
    "method:"search.getAllPagesInRange" ,
    "params" :[47.0681293,15.460332,2000,1]
}
```

Listing 2: JSON-RPC-Beispiel-Request

Der zweite Teil des hier angeführten Wertes beschreibt den Namen der aufzurufenden Methode dieses Objekts. Die Werte unter dem angeführten Parameter „params“ werden als Übergabeparameter an die aufgerufene Methode mitgegeben. In diesem speziellen Fall handelt es sich hierbei um den Breitengrad, Längengrad, Radius und um die maximale Anzahl an Artikel die der Client erwartet. Die Reihenfolge in der die Werte angegeben werden muss mit der Deklaration der Methode übereinstimmen. Nachdem der Request erfolgreich am Server eingegangen ist, werden die eintreffenden Daten auf eine passende Methode gemappt. Schlägt das Mapping fehl, da es keine Übereinstimmung zwischen Requestparamteren und vorhandenen deklarierten Funktionen gibt, wird eine Response mit dementsprechender Fehlermeldung generiert und an den Client zurückgeschickt. Im Falle eines erfolgreichen Mappings wird die so gefundene Methode, mit angegebenen Werten, aufgerufen und ausgeführt. Im beschriebenen Szenario handelt es sich um die Suche nach Artikeln in der Nähe. Die dafür zuständige Funktion verwendet die Werte Breitengrad und Längengrad um die aktuelle Position des Client zu bestimmen. Ausgehend von dieser wird die Distanz zwischen vorhandenen Artikeln, welche ebenfalls mit diesen Informationen versehen sind, berechnet. Anschließend wird noch der übergebene Wert des Radius herangezogen. Die zuvor kalkulierten Werte der Distanz müssen dabei jeweils kleiner als der angegebene Radius sein um für die Response in Frage zu kommen. Alle Artikel, die diese Voraussetzung erfüllen, werden in die Response, geordnet nach der eigentlichen Entfernung zum Client, aufgenommen und zum Client geschickt. Die maximale Anzahl von Artikeln, die auf diese Weise retourniert werden, ist einerseits limitiert durch den letzten Wert des „params“ Arrays und andererseits durch die

Anzahl der Ergebnisse der eigentlichen Suche.

2. Nach erfolgreicher Anfrage schickt der Server seine Antwort an den Client, zu sehen in Listing 3.


```
{
  "result": {
    "javaClass": "java.util.ArrayList",
    "list": [
      {
        "javaClass": "java.util.HashMap",
        "map": {
          "license": {
            "javaClass": "java.util.HashMap",
            "map": {
              "css": "ccbysa",
              "id": "CCBYSA30",
              "title": "CC BY-SA 3.0",
              "url": "https://creativecommons.org/licenses/by-sa
                  /3.0/"
            }
          },
          "distance": 537,
          "page": "AustriaWiki/Herz-Jesu-Kirche_(Graz)",
          "title": "Herz-Jesu-Kirche (Graz)",
          "url": "http://austria-forum.org/af/AustriaWiki
                                  /Herz-Jesu-Kirche_%28Graz%29"
        }
      }
    ]
  },
  "id": 10004
}
```


Listing 3: JSON-RPC-Beispiel-Response

Im weiteren Verlauf obliegt es dann der Verantwortung des Clients die Antwort richtig zu parsen und die erhaltene Information zu nutzen.
Wie hier ebenfalls zu erkennen ist, handelt es sich bei dieser Antwort um die JSON-RPC Spezifikation 1.0. Im Parameter „result“ ist die Antwort mit allen geforderten Inhalten verpackt, der Parameter „id“ ist, wie oben schon erwähnt, für die Zuord-

nung der Antwort an die zuvor gesendete Anfrage verantwortlich. In diesem speziellen Fall befindet sich in der Antwort der serialisierte Wert einer ArrayList. HashMaps bilden dabei die Einträge. Jede HashMap beschreibt einen gefundenen Artikel mit zusätzlichen für den Client relevanten Informationen. Darunter fallen Auskunft über dessen Lizenz, Entfernung zum gesendeten Standpunkt, Titel des Artikels, Name der JSPWiki Seite sowie die URL, die für die Anzeige des Artikels aufzurufen ist.

Für eine detaillierte Beschreibung aller Erweiterungen des SearchManagers und der JSONSearch Klasse wird an dieser Stelle an die angehängte Dokumentation der neuen Webservices verwiesen.

3.3.2.2 Technische Umsetzung der Informationsbeschaffung auf Clientseite

iOS

Für den iOS basierenden Prototypen ist eine Singleton-Klasse mit dem Namen „RequestManager“ erstellt worden. Diese ist zuständig für die Erzeugung der einzelnen Request-Objekte und für das Abschicken sowie das Empfangen derer. Genauer gesagt, beschreiben die jeweiligen Methoden der Klasse die bestimmten Features des Prototypen und ermöglichen es diese mit einem Funktionsaufruf durchzuführen. Dafür werden in den Methoden die spezifischen Requests zusammengebaut und an die verwendete Bibliothek Alamofire[63] weitergegeben. Diese ist schlussendlich zuständig für das eigentliche Absenden des Requests sowie für das Entgegennehmen der Response. So gesehen kann der RequestManager im weiterem Sinne auch als Fassade bezeichnet werden. Der RequestManager sowie die dazugehörigen Request-Klassen wurden in einer Art und Weise konzipiert, dass diese frei von Abhängigkeiten gegenüber den anderen Teilen des Programms sind, die nichts mit der Server-Client Kommunikation zu tun haben. Dies sorgt auch für eine gute Erweiterbarkeit, sodass, wenn in Zukunft neue Schnittstellen vom Webservice zu Verfügung gestellt werden, schnell und unkompliziert auf diese reagiert werden kann. Schlussendlich obliegt es dann den Benutzerinnen und Benutzer ein neues Update zu installieren um Erweiterungen wahrzunehmen. Abbildung 24 verdeutlicht anhand eines Klassendiagramms die Verbindung zwischen den einzelnen Klassen und die Architektur

[63] `https://github.com/Alamofire/Alamofire` zuletzt aufgerufen am 11.11.2016 um 10:04

der Kommunikation mit dem Server. Zu sehen ist hier, dass die jeweiligen Controller das benötigte NetworkDelegation Protokoll implementieren müssen. Wird eine Anfrage an den RequestManager gestellt, erstellt dieser die jeweiligen Requests und benachrichtigt anschließend den aufrufenden Controller mittels des bereitgestellten Protokolls über das Ergebnis. Zur Veranschaulichung wurde die Artikelsuche und deren Controller sowie die dazugehörige Request-Klasse gewählt.

Für die Realisierung der einzelnen Requests ist beschlossen worden, dass auf Ver-

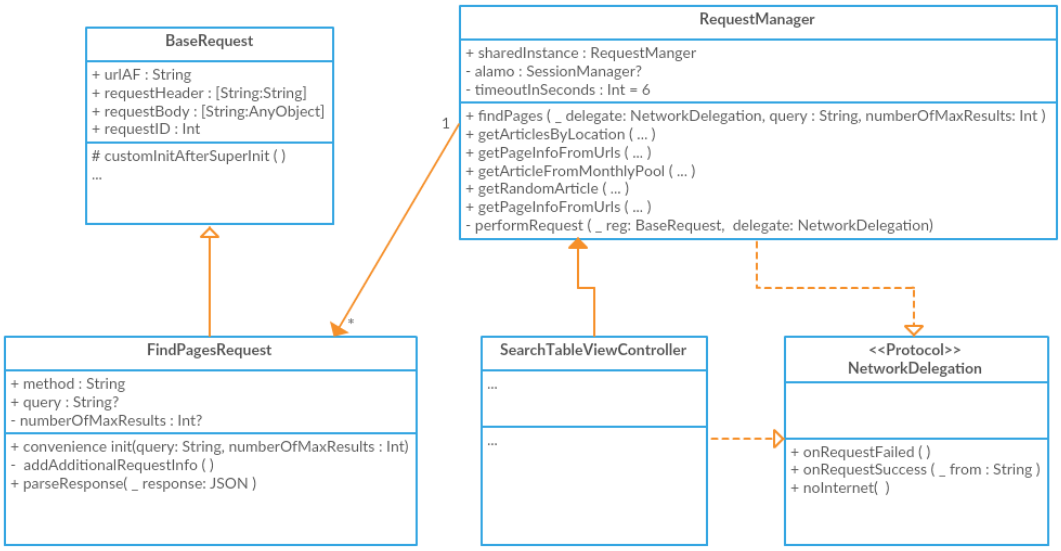

Abbildung 24: Klassendiagramm, Ablauf eines Server-Calls

erbung gesetzt wird. Grund dafür ist, dass sich die einzelnen Requests gewisse Felder im Header teilen, Ähnlichkeiten in der Struktur des Bodies aufweisen sowie denselben Endpunkt haben. Dementsprechend wurde eine Basisklasse mit dem Namen „BaseRequest“ implementiert. Ausgehend von dieser Basisklasse wurden alle benötigten spezifischen Request-Klassen erstellt, welche vom BaseRequest abgeleitet sind. Im RequestManger werden die spezifischen Request Objekte instanziiert und der ausführenden Funktion, `performRequest(_ reg: BaseRequest, delegate: NetworkDelegation )`, übergeben. Die eben genannte Funktion, in der schlussendlich der Request abgeschickt wird, extrahiert alle relevanten Informationen und übergibt diese der Bibliothek Alamofire.

Als Schnittstelle zwischen RequestManager und dem aufrufenden Controller dient, wie weiter oben schon angesprochen, ein Delegate-Objekt, welches beim Aufruf einer Methode des RequestManagers mitübergeben wird. Zusätzlich müssen je nach Auf-

gabe extra Parameter mitübergeben werden. Beim Aufruf der Artikelsuche wird beispielsweise die Suchanfrage sowie die maximale gewünschte Anzahl an gefundenen Artikeln zusätzlich zum Delegate Objekt der Funktion übergeben. Das Delegate-Objekt ist notwendig um den jeweiligen Controller mitzuteilen, ob der von ihm gestartete Request erfolgreich war oder fehlgeschlagen ist. Als Beispiel für eine Methode aus dem RequestManager ist in Listing 4 die Methode für das Suchen von Artikeln angeführt.

```
func findPages(_ delegate: NetworkDelegation, query : String,
    numberOfMaxResults: Int ){
        let findPagesReq : FindPagesRequest = FindPagesRequest(query:
            query, numberOfMaxResults: numberOfMaxResults)
        performRequest(findPagesReq, delegate: delegate)
    }
}
```

Listing 4: Artikelsuche

Das Parsing der einzelnen Responses wird in den spezifischen Request-Klassen behandelt. Hierfür delegiert Alamofire die Response an das jeweilige Request Objekt. Dieses ist im weiterem Verlauf dann zuständig für das korrekte parsing und speichert das Ergebnis in die jeweiligen Model-Klassen. Das Parsing geschieht grundsätzlich vor der Verständigung der Controller, ob dessen Request erfolgreich war oder nicht. Deswegen kann der jeweilige Controller bei Erfolg davon ausgehen, dass die Daten bereits verfügbar und einsatzbereit sind. Im Falle eines Misserfolgs wissen die jeweiligen Controller, dass keine Daten vorhanden sind und kümmern sich um etwaige Fehlermeldungen oder Hinweise für die Benutzerinnen und Benutzer. Damit auf das Parsing nicht vergessen wird, muss jede vom BaseRequest abgeleitete Klasse die Parsing Methode überschreiben. Wird diese nicht überschrieben, dann wird die Implementierung der Basisklasse ausgeführt. In dieser wird lediglich ein „fatalError“ mit dem Hinweis, dass diese zu überschreiben ist, ausgeführt. Da Swift das Konzept der Abstraktion nicht kennt, wurde die Methode des fatalErrors gewählt.[64]

Android

Für Android wurde im Grunde das gleiche Architektur Konzept gewählt, wie es für iOS gemacht worden ist. Dies liegt daran, dass iOS in der Entwicklung stets wei-

[64] `http://stackoverflow.com/a/24110406` zuletzt aufgerufen am 11.01.2017 um 17:24

ter fortgeschritten war. Die zentrale Logik für das Erstellen der einzelnen Requests und die Weiterleitung an die verwendete externe Bibliothek wird ebenfalls vom „RequestManager“, implementiert in Java für Android, verwaltet und ausgeführt. Für Android wurde Android Asynchronous Http Client[65] als Bibliothek gewählt. Bei den Request-Klassen des Android Prototypen wird ebenfalls aus dem gleichen Grund wie bei iOS auf Vererbung gesetzt. Da in Java im Gegensatz zu Swift allerdings das Konzept der Abstraktion durchaus bekannt ist, handelt es sich bei der Basisklasse der Requests um eine abstrakte Basisklasse.
Die Architektur der Kommunikation zwischen dem RequestManager und dem restlichen Programm ist ebenfalls ähnlich aufgebaut wie bei der iOS-Variante. Jedoch wurden die Java spezifischen Eigenschaften berücksichtigt. Als Schnittstelle wird hier ein Interface verwendet über welches Informationen bzgl. Erfolg und Misserfolg der einzelnen Requests gesendet werden. Der sequentielle Ablauf ist ebenfalls gleich gehalten. Bei Erfolg wird zuerst die eintreffende Response behandelt und in den jeweiligen Model-Klassen gespeichert. Anschließend wird die jeweilige Activity benachrichtigt.

3.3.3 Umsetzung der ausgearbeiteten User-Stories

Für die technische Umsetzung des Prototypen wurde der in Kapitel Stand der Technik beschriebene Ansatz einer hybriden Applikation gewählt. Der Prototyp ist dahingehend als hybride zu bezeichnen, da die präsentierten Inhalte über das Laden einer URL des Austria-Forums angezeigt werden. Neben dieser Art der Visualisierung, die einem Container ähnelt, wurde mit den durch eine native Umsetzung zur Verfügung stehenden Mitteln der Zugriff auf die neu entwickelten Funktionen der Austria-Forum API ermöglicht. Zusätzlich dazu wird somit auch der Weg zu den internen Sensoren und Services, wie beispielsweise Benachrichtigungen und Lokalisierung, geebnet. Ebenfalls wird für das Anzeigen der Inhalte ein eigener Skin verwendet. Dieser Skin zeichnet sich dadurch er, dass lediglich der Inhalt eines Artikels angezeigt wird. Sämtliche Bedienelemente, Navigation sowie Header und Footer der eigentlichen Austria-Forum Webseite sind dabei ausgeblendet. Auf Grund all dieser Gegebenheiten wurde sozusagen ein Austria-Forum-Reader entwickelt. Dass die im Reader angezeigten Inhalte ohne weiteres in einer der Bildschirme angepassten Form und Größe angezeigt werden, ist der Webseite des Austria-Forums, welche sich durch

[65] `http://loopj.com/android-async-http/` zuletzt aufgerufen am 11.01.2017 um 21:09

ein responsives Webdesign auszeichnet, zu verdanken.
Die zuvor durchgeführte Auswertung der Evaluierung und die daraus entstandenen User-Stories bilden die Basis des Anforderungsprofils für den resultierenden Prototypen. In weiterer Folge wird nun etwas genauer auf dieses eingegangen und in Anbetracht dessen die technische Umsetzung beschrieben.

Zufälliger Artikel:
Eine anwendende Person möchte gerne etwas Unbekanntes lesen um damit neues Wissen zu generieren, hat aber keine Idee, wonach sie suchen soll.

Wie aus der User-Story hervorgeht, geht es hierbei um Informationsgewinnung für Benutzerinnen und Benutzer. Dabei soll, wie zuvor schon erwähnt, der Aufwand sehr gering bleiben. Dies wurde dahingehend berücksichtigt, dass für die Anwendung dieses Features lediglich eine Interaktion durchgeführt werden muss. Wie auf Abbildung 25 ersichtlich ist, ist die Funktionalität des zufälligen Artikels (2) zentral durch eine Berührung ausführbar. Für die Möglichkeit der Modifizierung des Resultats und um somit den absoluten Zufallsfaktor zu reduzieren, wurde beschlossen, dass Benutzerinnen und Benutzer die Kategorie, welche für die Ermittlung eines zufälligen Artikels herangezogen wird, ändern können. Die Kategorie kann in den Einstellungen der Applikation angepasst werden. Diese sind durch (6), zu sehen auf Abbildung 25, erreichbar. Wird an dieser Stelle eine Kategorie, wie auf Abbildung 26 (2.1) zu sehen ist, ausgewählt, so wird diese beim nächsten Aufruf des zufälligen Artikels berücksichtigt. Genauer gesagt, wird durch die getätigte Auswahl der Request des zufälligen Artikels dahingehend ergänzt, dass zusätzlich die Kategorie mitgesendet wird. Die Logik des Webservice verwendet dann in weiterer Folge die mitgesendete Kategorie und wählt anhand dessen einen zufälligen Artikel aus. Dieser wird dann anschließend wieder an den Client zurückgeschickt. Befindet sich im Request keine Kategorie, so wird standardmäßig die komplette Wissensbasis für die Auswahl des Artikels berücksichtigt. Benutzerinnen und Benutzer können dabei jederzeit je nach Interesse zwischen den Kategorien wechseln und erhalten somit einen angepassten zufälligen Artikel, welcher, wenn aktiviert, aus einem Bereich kommt der von Interesse ist.

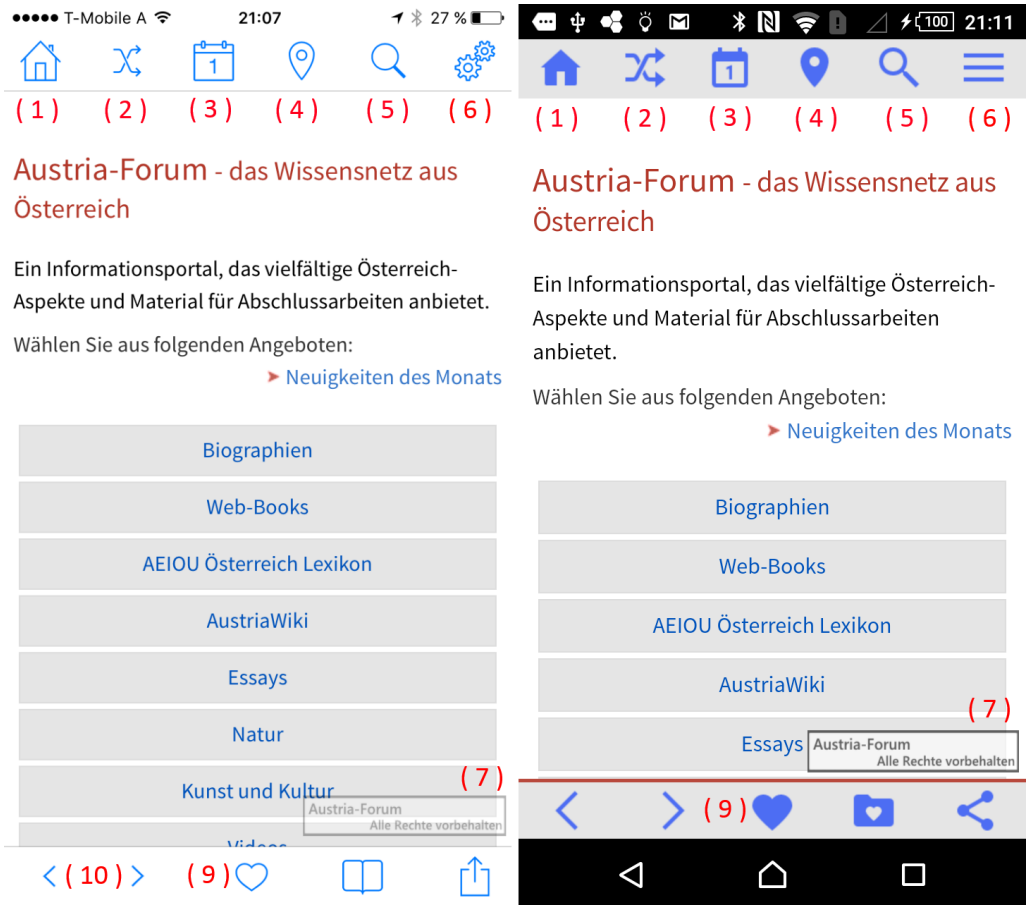

Abbildung 25: Startscreen des Prototypen, iOS / Android

Artikel des Monats:

Benutzerinnen und Benutzer möchten gerne etwas Neues, was aktuell in diesem Monat von Bedeutung ist, in Erfahrung bringen um zeitnahe noch unbekannte Information zu erhalten.

Ähnlich wie beim zufälligen Artikel ist die Funktionalität des Artikels des Monats mit nur einer Berührung durchführbar. Auf Abbildung 25 (3) ist diese ersichtlich. Zu erkennen ist des Weiteren, dass auch dieses Feature zentral gelegen und ohne viel Aufwand abrufbar ist. Die Umsetzung ist dabei wie folgt: Führen Benutzerinnen und Benutzer diese Aktion durch, so wird vom Server ein zufälliger Artikel aus allen in diesem Monat neu erstellten Inhalten ausgewählt und dem Client als Antwort gesendet. Der zurückgegebene Artikel wird dann in weiterer Folge lokal auf dem Gerät gespeichert. Wenn für den aktuellen Monat bereits ein gespeicherter Artikel

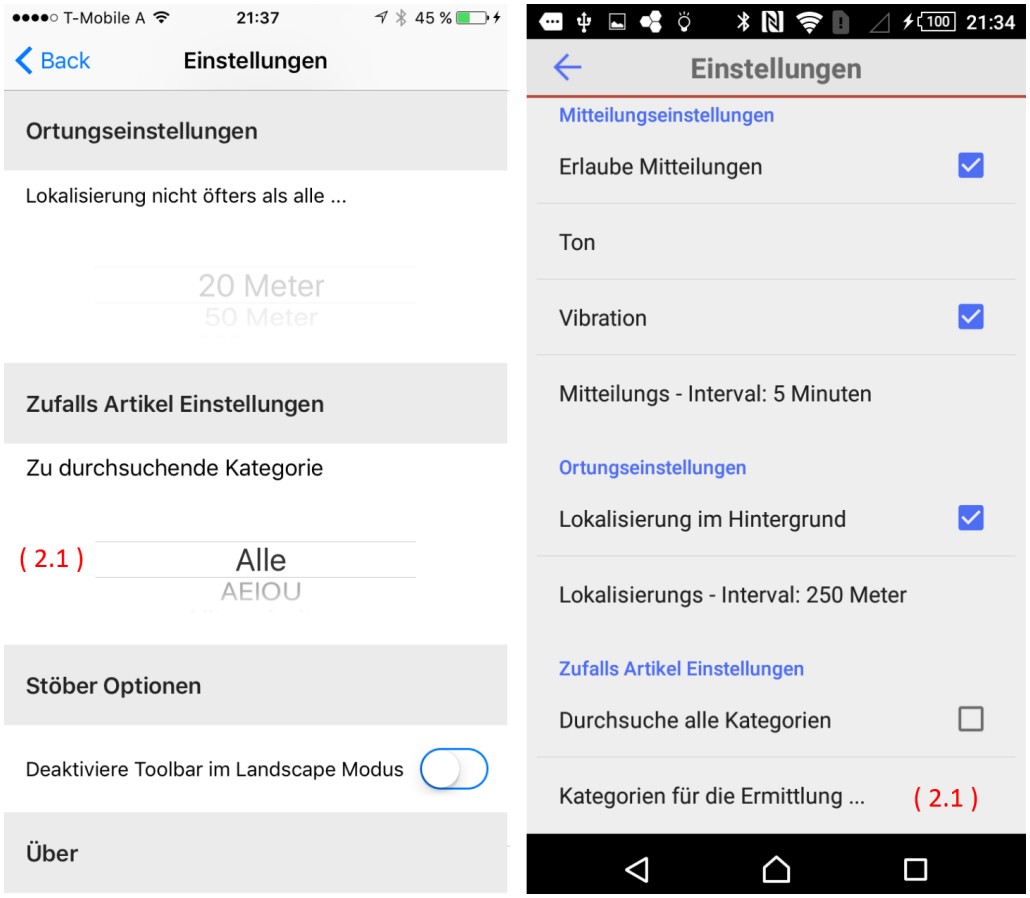

Abbildung 26: Einstellungen Teil 2, iOS / Android

vorliegt, so wird dieser geladen und keine weitere Anfrage zum Server geschickt. Zusätzlich wurde für die Zukunft bedacht, dass es möglich sein soll einen bestimmten Monat auszuwählen, bevor der Request abgeschickt wird. In diesem Fall müsste der Request dahingehend modifiziert werden, dass ein bestimmter Monat und ein bestimmtes Jahr mitgesendet wird. Der Artikel des Monats wird dann unter Berücksichtigung der mitgelieferten Daten ausgewählt. Diese Möglichkeit wurde zwar serverseitig berücksichtigt, findet aber noch keine Anwendung in den einzelnen mobilen Applikationen.

Artikel in der Nähe – Aktiv:
Benutzerinnen und Benutzer befinden sich an irgendeinem Ort und wollen wissen, was für relevante Artikel es in ihrer Nähe gibt.

Wie auch schon bei den anderen Features angesprochen, setzt auch jenes der aktiven Abfrage von Artikeln in der Nähe darauf möglichst wenig Aufwand für Benutzerinnen und Benutzer zu verursachen. Betätigen Benutzerinnen und Benutzer das Lokalisierungssymbol, zu sehen in Abbildung 25 (4), dann wird eine Listenansicht mit Artikeln geladen. Bei den Listeneinträgen handelt es sich grundsätzlich um Artikel, die nach Entfernung zur aktuellen Position der Benutzerinnen und Benutzer geordnet sind. Ein Beispiel solch einer Liste ist in Abbildung 27 zu sehen. Als weitere Information wird die genaue Entfernung und der Titel des Artikels angegeben. Befindet sich ein Artikel weniger als 1000 Meter entfernt so wird die Distanz in Meter angegeben. Überschreitet dieser Wert einen Kilometer, dann wird die Distanz in Kilometer, gerundet auf zwei Nachkommastellen, angegeben. Die Anzahl der Artikel wird grundsätzlich durch zwei Werte beeinflusst, welche im dafür verantwortlichen Request mitgeschickt werden. Bei diesen Werten handelt es sich zum einen um die maximale Anzahl von Artikeln, die angefordert werden, und zum anderen um den Radius, der für die Suche einzubeziehen ist. Der Radius wird dabei ausgehend von der aktuellen Position der Benutzerinnen und Benutzer gezogen. Je nachdem, ob und wie viele Ergebnisse der Server bei seinen Berechnungen findet, wird keiner oder maximal so viele wie zuvor angefordert, zurückgesendet. Öffnen Benutzerinnen und Benutzer zum ersten Mal diese Ansicht, dann werden sie, bevor der Request erfolgreich abgesetzt werden kann, gefragt, ob die benötigten Berechtigungen um auf den aktuellen Standpunkt zugreifen zu dürfen, freigegeben werden. Wenn auf dem aktuellen Gerät allerdings eine Android Version installiert ist, die unter der API Version 23 liegt, dann wird an dieser Stelle nicht nach der Bewilligung für die benötigten Berechtigungen verlangt. Denn wie schon an anderer Stelle erwähnt, werden bei diesen Versionen alle geforderten Rechte bereits bei Installation der Applikation abgefragt. Wird dem nicht zugestimmt, kann die Applikation nicht installiert werden. Bei neueren Versionen oder unter iOS können Benutzerinnen und Benutzer natürlich die geforderten Berechtigungen verweigern. Ist dies der Fall, dann funktioniert die Suche nach Artikel in der Nähe nicht. Einmal abgelehnt, müssen Benutzerinnen und Benutzer in den Systemeinstellungen der jeweiligen Betriebssysteme der Erlaubnis für die Benutzung des aktuellen Standortes wieder zustimmen, sodass die Suche nach Artikeln in der Nähe wieder ordnungsgemäß arbeiten kann. Da der Weg dorthin nicht leicht zu finden ist, wurde beschlossen eine Hilfestellung beim Ein- und Ausschalten der Lokalisierung, in der Applikation selbst, zu geben.

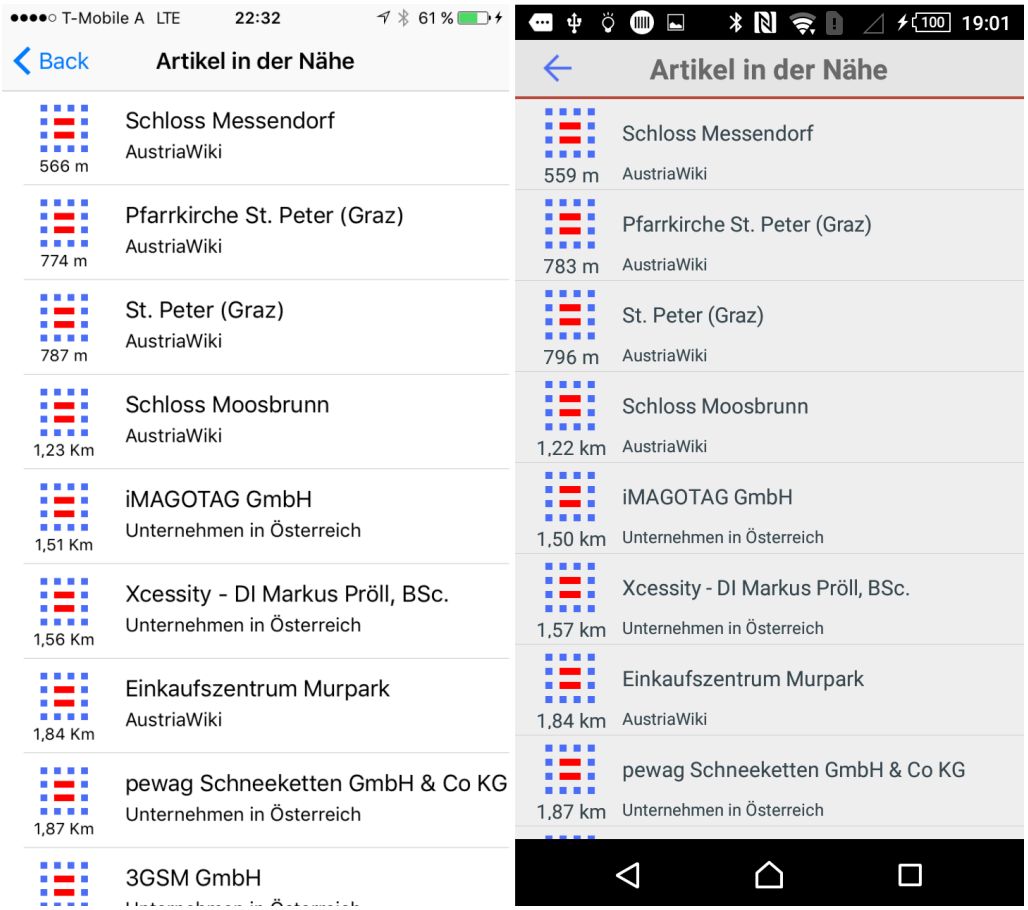

Abbildung 27: Artikel in der Nähe - Aktiv, iOS / Android

Wenn Benutzerinnen und Benutzer nachträglich in den Einstellungen der Applikation der Benutzung des Standortes zustimmen, so werden diese darauf hingewiesen, dass zuerst die allgemeine Berechtigung in den Systemeinstellungen aktiviert werden muss. Wenn dieser Hinweis angezeigt wird, wird Benutzerinnen und Benutzern die Möglichkeit geboten sich auf Wunsch direkt in die dementsprechenden Einstellungen schicken zu lassen. Um den beschriebenen Ablauf des Abrufens von Artikel in der Nähe besser darzustellen wurde das in Abbildung 28 gezeigte Anwendungsfalldiagramm erstellt.

Artikel in der Nähe – Passiv:
Benutzerinnen und Benutzer sind in Bewegung und wollen laufend über Artikel in seiner/ ihrer Nähe benachrichtigt werden.

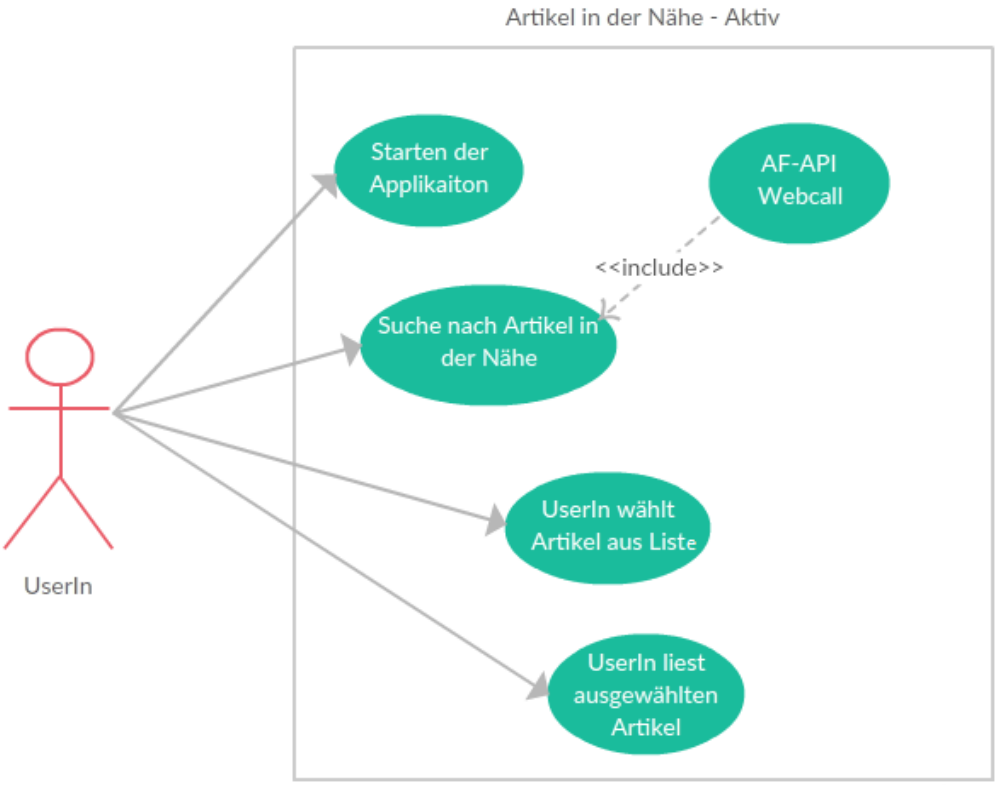

Abbildung 28: Anwendungsfalldiagramm, Artikel in der Nähe - Aktiv

Neben der aktiven Lösung um Artikel in der Nähe zu erhalten ist auch ein passiver Weg umgesetzt worden. Für dieses Szenario wurde das in Abbildung 29 gezeigte Anwendungsfalldiagramm erstellt, welches in weiterer Folge den Ablauf besser verdeutlichen soll. Um allerdings die passive Möglichkeit in Anspruch zu nehmen müssen Benutzerinnen und Benutzer diese zunächst in den Einstellungen der Applikation aktivieren. Als Werkseinstellung wurde beschlossen, dass diese zu Start deaktiviert ist. Obwohl durch dieses Feature passive Informationsbeschaffung bzw. Informationsaufnahme ermöglicht wird, darf nicht vergessen werden, dass auch kleinere Nachteile dadurch entstehen können. Denn um passiv auf relevante Artikel aufmerksam gemacht zu werden muss die Lokalisierung im Hintergrund laufen und stets den aktuellen Ort bekannt geben. Dies führt unter anderem natürlich zu einer zusätzlichen Belastung für den Akku eines Gerätes und sorgt für erhöhten Energieverbrauch. Die zusätzliche Energiebelastung sowie der Gedanke an die Privatsphäre waren ausschlaggebend, dass beschlossen worden ist, dass der passive Modus standardmäßig deaktiviert ist.

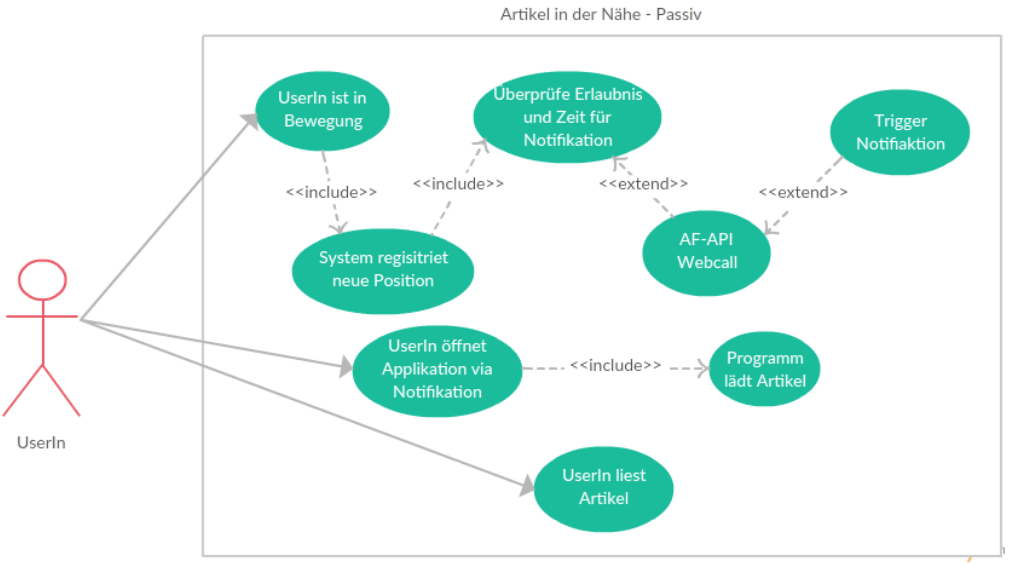

Abbildung 29: Anwendungsfalldiagramm, Artikel in der Nähe - Passiv

Grundsätzlich wird die Lokalisierung vom jeweiligen Betriebssystem verwaltet. Trotzdem sind Entwickler/ Entwicklerinnen angehalten, dass diese mit größter Vorsicht und den passenden Werten zu verwenden ist, sodass der benötigte Energieverbrauch auf ein Minimum reduziert wird. Um dies zu bewerkstelligen, werden von den jeweiligen Betriebssystemen vor Start der Lokalisierung gewisse Einstellungsmöglichkeiten angeboten. Mit Hilfe dieser kann die Lokalisierung dahingehend modifiziert werden, dass sie an die Bedürfnisse der jeweiligen Applikation angepasst wird. Dadurch wird quasi ein passendes Kosten-Nutzungsverhalten hergestellt und somit die Akkulaufzeit optimiert.
Einer dieser einzustellenden Werte betrifft die Genauigkeit der Lokalisierung. Es ist möglich eine gewisse Fehlertoleranz bei der Bestimmung des aktuellen Ortes zu erlauben. Grundsätzlich kann gesagt werden: Je genauer die Positionsbestimmung desto mehr Energie wird für das Ermitteln benötigt. Wenn eine Fehlertoleranz angegeben wird, wird auch weniger Energie verbraucht und schont gleichzeitig den Akku des Gerätes. Hier ist dementsprechend der für die Applikation passende Wert anzugeben um eine möglichst effiziente Verwendung der Ressourcen zu garantieren. Etwaige Fehlertoleranzen können zwischen 10-100 Meter oder gar sehr grob mit Abweichungen im Kilometerbereich liegen[66].

[66] `https://developer.apple.com/reference/corelocation/cllocationmanager/1423836-desiredaccuracy`

Ein weiterer Wert, der für die Optimierung angegeben werden kann, ist die Distanz, die zurückgelegt werden muss, bevor ein Positionswechsel als solcher erkannt werden soll. Hier ist zuvor noch zu erwähnen, dass, wenn ein Lokalisierungsdienst gestartet wird, dieser je nach Einstellung regelmäßig Information über die aktuelle Position liefert. Wie oft dies geschieht hängt von der zuvor erwähnten Einstellung der zurückzulegenden Distanz ab. Wird die Distanz mit 0 angeben, wird die Applikation durchgehend mit Positionsupdates beliefert, auch wenn Benutzerinnen und Benutzer gar nicht in Bewegung sind. Wird allerdings ein Wert größer als 0 angegeben, so werden Updates mit neuen Werten nur dann rausgegeben, wenn diese Distanz tatsächlich zurückgelegt worden ist[67].
Die Konfiguration der zwei oben genannten Faktoren beeinflusst dementsprechend den Energieverbrauch der Applikation zu großen Teilen. Auf Grund dessen wurde die bestmöglichste Vorgehensweise dafür diskutiert und umgesetzt.
Allerdings haben verschiedene Benutzerinnen und Benutzer unterschiedliche Meinungen über die für sie beste Konfiguration. Für manche reicht es, wenn sie sporadisch über nahegelegene Artikel informiert werden. Dafür soll die Applikation aber möglichst wenig Energie verbrauchen. Andere hingegen nehmen gerne einen höheren Energieverbrauch in Kauf, wenn sie dadurch möglichst oft über Interessantes in ihrer Nähe benachrichtigt werden. Um beide Extreme sowie auch ein Mittelmaß zufrieden zu stellen ist beschlossen worden, Benutzerinnen und Benutzer eine Modifikation der Lokalisierung zu ermöglichen, sodass sie je nach aktuellem Bedürfnis die Konfiguration anpassen können. Als variabel wurde allerdings nur der Wert der Distanz, die zurückgelegt werden muss, definiert. Im Sinne eines niedrigen Energiebedarfs ist die Genauigkeit der Positionsbestimmung trotzdem als Konstante definiert worden.
Umgesetzt wurde dies durch die Bereitstellung eines änderbaren Wertes in den Einstellungen der Applikation. Die in Abbildung 30 (4.1) zu sehende Einstellungsmöglichkeit beschreibt dabei die Distanz, die zurückzulegen ist, bevor ein Positionsupdate ausgesendet wird.
Bei der Änderung von diesem Wert wird anschließend der Lokalisierungsdienst unter

`https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest#setPriority(int)` jeweils zuletzt aufgerufen am 29.01.2017 um 16:55

[67] `https://developer.apple.com/reference/corelocation/cllocationmanager/1423500-distancefilter`
`https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest.html#setSmallestDisplacement(float)` jeweils zuletzt aufgerufen am 29.01.2017 um 17:04

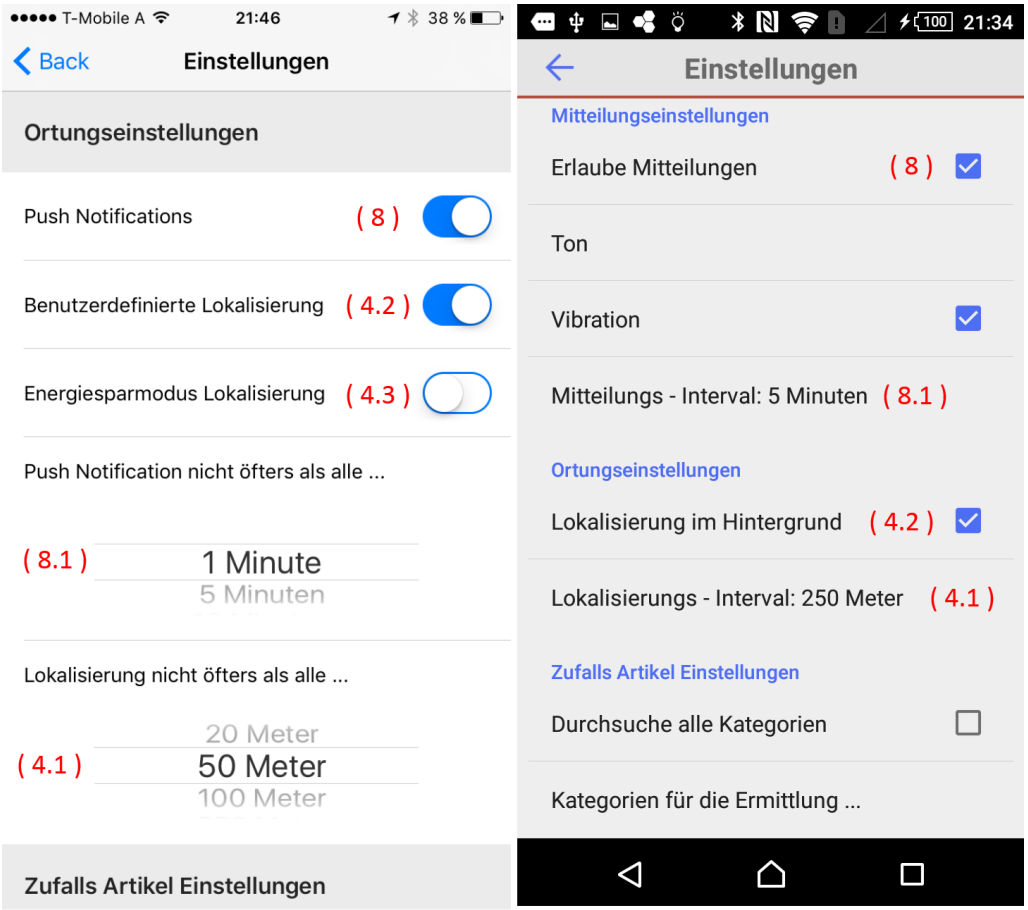

Abbildung 30: Einstellungen Teil 1, iOS / Android

Berücksichtigung der eben geänderten Variable neu gestartet. So können Benutzerinnen und Benutzer jederzeit das Verhalten der Applikation beispielsweise an ihre Bewegungsgeschwindigkeit anpassen. Zusätzlich wurde die Entscheidung getroffen Benutzerinnen und Benutzer ebenfalls die Möglichkeit zu bieten die Lokalisierung komplett auszuschalten. Und das ohne den Umweg über die Systemeinstellungen der jeweiligen Betriebssysteme gehen zu müssen. Für diesen Fall wurden ebenfalls Schalter in den Einstellungen der jeweiligen Applikation eingebaut, welche auf Abbildung 30 (4.2, 4.3) zu sehen sind.
Neben der Lokalisierung an sich wird, um das volle Potential des passiven Modus auszuschöpfen, auch das Benachrichtigungssystem der jeweiligen Betriebssysteme verwendet. Genauer gesagt wurden für die Umsetzung lokale Benachrichtigungen eingesetzt. Im Gegensatz zu „remote push notifications“ können lokale Benachrich-

tigungen direkt am Gerät erstellt und ausgelöst werden. Damit eine Applikation auf iOS generell lokale Benachrichtigungen aussenden kann, muss zuerst die Berechtigung dafür abgefragt und erteilt werden. Fehlt an dieser Stelle die Zustimmung zur Erlaubnis, kann die Applikation keine Benachrichtigungen aussenden. Auf Android Basis ist diese Erlaubnis nicht erforderlich.
Wie in Kapitel 1.0 zuvor schon erwähnt, muss die Verwendung von Benachrichtigungen mit Bedacht durchgeführt werden, damit Benutzerinnen und Benutzer nicht in Gefahr geraten zu denken, dass sie zu häufig benachrichtigt werden. Aufgrund dessen wurde beschlossen, dass Benutzerinnen und Benutzer die Häufigkeit sowie den generellen Empfang in der Applikation direkt einstellen können. Abbildung 30 (8) zeigt die Möglichkeit den generellen Empfang ein- und auszuschalten. Dadurch soll ein Gefühl der Kontrolle über das Verhalten der Applikation vermittelt werden, sodass eine geglaubte, zu häufige Benachrichtigung nicht in einer negativen Einstufung des Prototypen resultiert, sondern in der Aktion die aktuellen Einstellungen zu ändern. Der in Abbildung 30 (8.1) zu sehende Wert kann von Benutzerinnen und Benutzer verändert werden und wird bei einer mitgeteilten Ortsänderung zusätzlich berücksichtigt. Im Sinne des Energiesparens wird, sobald vom Lokalisierungsdienst ein neues Positionsupdate rausgegeben wird, zuerst überprüft, ob das geforderte Zeitintervall der letzten Benachrichtigung überschritten worden ist. Ist dies der Fall, wird eine weitere Bearbeitung der neuen Position durchgeführt. Wenn aber die gesetzte Zeitmenge seit der letzten Benachrichtigung nicht überschritten worden ist, dann wird auch keine Anfrage zum Server geschickt. Ist die benötigte Zeit vergangen, wird die aktuelle Position, wie weiter oben beschrieben, an den Server geschickt und eine den Umständen entsprechende Antwort zurückgeliefert. Je nach Antwort wird eine lokale Benachrichtigung erstellt und ausgeschickt.

Für die Implementierung des passiven Modus und des zuständigen Benachrichtigungssystem wurde in der Android Version ein Service[68] entwickelt. Dieser Service baut eine Verbindung zu Google Play Services Location APIs[69] auf und ist dementsprechend verantwortlich für die Positionsbestimmung und dessen Veränderung. Beim Verbindungsaufbau wird der variable Wert der Distanz, die zurückzulegen ist,

[68] `https://developer.android.com/reference/android/app/Service.html` zuletzt aufgerufen am 12.09.2016 at 08:39

[69] `https://developers.google.com/android/reference/com/google/android/gms/location/package-summary` zuletzt aufgerufen am 12.09.2016 at 08:45

sowie die Genauigkeit berücksichtigt. Einmal gestartet läuft dieser Service solange im Hintergrund weiter, bis er gestoppt wird. Für die iOS Version wurde eine Fassade implementiert, welche den Zugriff auf den CLLocationManger[70] vereinfacht. Der CLLocationManger ist unter iOS zuständig für die Bestimmung der Position sowie die Registrierung einer Ortsveränderung und bietet generell zwei Arten der Lokalisierung an.
Grundsätzlich unterscheiden sich diese durch die Konfigurationsmöglichkeiten. Bei der Ersten kann ähnlich wie bei Android die Genauigkeit der Lokalisierung sowie die Distanz, die überwunden werden muss, angegeben werden. Bei der Zweiten ist es nicht möglich die angesprochenen Werte anzugeben. Vielmehr stellt diese quasi einen Energiesparmodus des internen Lokalisierungsdienstes dar. In diesem Fall wird ein Positionsupdate ausgesendet, sofern mindestens 500 Meter zurückgelegt worden sind. Positionsupdates durch diese Variante dürfen sogar die Applikation starten, wenn diese zuvor beendet worden ist. Wird die Applikation auf diesem Wege gestartet, hat sie ein Zeitfenster von ca. 10 Sekunden um etwaige Operationen durchzuführen[71].
In dem resultierenden Prototypen für iOS werden beide dieser Möglichkeiten angeboten. Entscheiden sich Benutzerinnen und Benutzer für erstere Variante, läuft der Lokalisierungsdienst im Hintergrund und versorgt die Applikation je nach Einstellungen mit Positionsupdates. Diese Updates werden ausgesendet, sofern Benutzerinnen und Benutzer in Bewegung sind. Registriert das Gerät 15 Minuten lang keine Änderung der Position, dann wird der Lokalisierungsdienst vom Betriebssystem gestoppt. Durch diesen von iOS zu Verfügung gestellten Mechanismus wird zusätzlich unnötiger Energieverbrauch minimiert.

Textbasierte Suche

Benutzerinnen und Benutzer würden gerne zu einem für sie relevantem Thema oder Stichwort zusätzliche Information und Artikel erhalten.

Die textbasierte Suche ist ebenfalls einfach von der Toolbar aus erreichbar. Durch ei-

[70] `https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CLLocationManager_Class/` zuletzt aufgerufen am 12.09.2016 at 08:53

[71] `https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/LocationAwarenessPG/CoreLocation/CoreLocation.html#//apple_ref/doc/uid/TP40009497-CH2-SW9` zuletzt aufgerufen am 14.01.2017 um 15:07

ne Berührung auf das Such-Symbol, welches in Abbildung 25 (5) ersichtlich ist, wird ein neuer Bildschirm geladen, welcher eine Suchleiste beinhaltet. Die Suche wurde dabei als Echtzeitsuche umgesetzt. Dementsprechend werden bei jeder Änderung neue Ergebnisse geladen. Die vom Server gelieferten Ergebnisse werden in einer Liste, geordnet nach einem „Score“, angezeigt. Dieser sogenannte Score wird serverseitig berechnet und beschreibt die Relevanz eines Artikels zum gegebenen Suchinhalt. Je höher dieser Wert ist, desto mehr Relevanz hat ein Artikel zur aktuellen Sucheanfrage und wird deswegen an oberster Stelle angezeigt. Als überblicksmäßige Beschreibung des Resultats wird der Titel des Artikels sowie die Kategorie, welcher der Artikel angehört, angezeigt. Zusätzlich wird, sofern vorhanden, ein Bild des aktuellen Artikels angezeigt.

Lizenz Information über angezeigte Artikel

Die Benutzerinnen und Benutzer würden gerne wissen unter welcher Lizenz der gerade aktive Artikel steht, damit sie über die Verwendung des Inhaltes Bescheid wissen.

Wie zuvor schon erwähnt, wurde dieser Punkt ebenfalls mit einer hohen Priorität eingestuft. Inspiriert durch Hammelehle (2013) und Weatherhead (2014), die darüber geschrieben haben, dass heutzutage alles wirklich schnell ablaufen und verfügbar sein muss, wurde für die Umsetzung beschlossen den Grundgedanken eines Floating Action Button[72] umzusetzen. Dadurch ist es für Benutzerinnen und Benutzer möglich in kürzester Zeit festzustellen, um was für eine Lizenz es sich handelt. Dies wurde bei beiden Versionen des Prototypen mit Hilfe einer transparenten View, welche rechts unten in der Ecke platziert wurde, realisiert. Diese sind auf auf Abbildung 25 (7) zu sehen. Die Information über die Lizenzierung wird vom Server bereitgestellt. Diese wird direkt bei Informationsbeschaffung in der Response mit einem Artikel mitgeliefert. Dadurch werden jene Fälle abgedeckt, bei denen die eigens entwickelten Webservices und Features aufgerufen werden. Wenn Benutzerinnen und Benutzer allerdings direkt durch die geladenen Artikel mit Hilfe der darin vorkommenden Links navigieren, wird im Hintergrund und, ohne dass sie dadurch gestört werden, explizit ein Request abgesetzt, welcher Informationen über die Lizenz des aktuellen Artikels

[72] `https://material.google.com/components/buttons-floating-action-button.html` zuletzt aufgerufen am 10.11.2016 um 17:07

anfordert. Die angezeigte View wird dann je nach Art der Lizenz mit dem passenden Logo befüllt, sodass möglichst schnell ersichtlich ist, um welche Art der Lizenz es sich handelt. Bei Berührung auf die Lizenzinformation wird ein dazu passender Link in einem externen Browser geladen und angezeigt. Ersichtlich ist hier die genaue Beschreibung der aktuellen Lizenz.
An dieser Stelle wurde bewusst die Entscheidung getroffen, dass die Lizenzinformationen nicht direkt im Prototypen geladen werden, da es sich bei den meisten Links um externe Webseiten handelt. Links sowie verwendete Grafiken für das Handling der Lizenzen können auf der Austria-Forum Webseite eingesehen werden.[73]

Informationsbeschaffung

Benutzerinnen und Benutzer möchten nach Erstinstallation gleich und ohne Umwege den vollen Funktionsumfang der Applikation genießen.

Wie schon zuvor erwähnt, ist im Rahmen etwaiger Diskussionen durch Expertenrunden und auf Grund der gegebenen Serverarchitektur beschlossen worden, dass für die Informationsbeschaffung ein Weg gewählt wird, bei dem die Benutzerinnen und Benutzer zwar über eine gültige Verbindung zum Internet verfügen müssen, dafür aber ohne großes Warten direkt mit der Applikation arbeiten können. Für jedes der bereits angesprochenen eigens implementierten Features wurde serverseitig eine neue Methode implementiert, die durch das JSON-RPC ansteuerbar ist. Für die entstandenen Prototypen ging es nicht primär um die Client-Server-Kommunikation, obwohl ohne eine solche die Prototypen nicht funktionieren würden. Vielmehr wird diese als gegeben angesehen und ist für die Umsetzung zu verwenden. Dennoch musste eine Kommunikation implementiert werden. Deswegen wurde für das Abschicken und das Empfangen von Requests und Responses zu und vom Server für beide Prototypen unterschiedliche, externe Bibliotheken verwendet. Diese sind allerdings, wie schon erwähnt, nur zuständig für Senden und Empfangen. Was wie gesendet wird und wie eine Antwort vom Server zu verarbeiten und zu interpretieren ist, ist Aufgabe der Prototypen. Die Programmarchitektur der Kommunikation wurde zuvor schon in Kapitel 3.3.2.2 beschrieben.

Lesezeichen

[73] `http://austria-forum.org/af/Lizenzen` zuletzt aufgerufen am 10.11.2016 um 17:26

Benutzerinnen und Benutzer möchten einen Artikel, den sie gerade lesen, als Lesezeichen speichern, damit sie auf diesen später schnell Zugriff haben.

Wie auch bei allen anderen aktiv zu betätigenden Features ist auch das Setzen von Lesezeichen durch nur eine Berührung zu realisieren. Die Möglichkeit wurde dabei so umgesetzt, dass sie von überall schnell und zentral durchführbar ist. Dafür wurde ein Icon in Form eines Herzes, siehe dazu Abbildung 25 (9), im User-Interface platziert. Dasselbe ist auch verantwortlich für das Entfernen eines bereits gesetzten Lesezeichens bzw. einen Artikel, der „geliked“ wurde. Die Lesezeichen werden dabei lokal persistiert und sind bis zur Deinstallation der Applikation oder bis zum manuellen Entfernen auf dem Gerät verfügbar. Für die Entwicklung wurde beschlossen die Persistenz der Daten durch das Schreiben in eine dafür erstellte Datei zu garantieren.
Die Android Version des Prototypen arbeitet dabei mit Serialisierung der Artikel-Objekte und schreibt diese dann anschließend getrennt durch einen vordefinierten Delimiter in die Datei. De-/Serialisiert werden die Objekte mit Hilfe der gson[74] Bibliothek.
Anders ist es bei der iOS-Version. Hier ist entschieden worden, dass die markierten Artikel in einer Property List Datei[75], kurz plist Datei gespeichert werden. Hierbei handelt es sich letztendlich um eine Datei, welche ihre Daten im XML-Format struktuiert.

Navigation

Benutzerinnen und Benutzer klicken auf einen verlinkten Artikel und würden gerne zurück navigieren um den zuvor aktiven Artikel weiterzulesen.

Die Navigation beruht im Grunde auf dem Navigationsstack der verwendeten Technologien. Wenn Benutzerinnen und Benutzer zurück oder nach vorne gehen wollen, müssen sie dafür das dementsprechende Icon des User-Interfaces betätigen. Abbildung 25 (10) zeigt die dafür notwendigen Aktionen. Grundsätzlich bieten die ver-

[74] `https://github.com/google/gson` zuletzt aufgerufen am 11.11.2016 um 12:55

[75] `https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Articles/AboutInformationPropertyListFiles.html` zuletzt aufgerufen am 11.11.2016 um 12:55

wendeten Technologien, sei es die WebView[76] für Android oder das WKWebKit[77] unter iOS, eine Möglichkeit an zurück- sowie vorwärts zu navigieren. Die Verwaltung des Navigationsstacks wird ebenfalls von diesen gehandhabt. Es müssen nur die dafür vorgesehenen Methoden aufgerufen werden. Diese werden dann durch das Betätigen der passenden Icons ausgeführt, sofern dies möglich ist.

Home:
Benutzerinnen und Benutzer möchten gerne wieder zurück zur Austria-Forum Startseite um neue Kategorien durch Navigieren zu erforschen.

Diese Story beschreibt das Laden des Startbildschirms. Wenn zu Beginn die Appliaktion gestartet wird, lädt sich in der vorgesehenen View die Startseite des Austria-Forums. Ausgehend von dieser ist es möglich per Navigation die verschiedenen Inhalte des Austria-Forums zu erkunden. Wird die Applikation beendet, dann wird bei Neustart der zuletzt geöffnete Artikel erneut geladen. Aus diesem Grund wurde der „Home-Button“ implementiert, sodass explorative Benutzerinnen und Benutzer stets und von überall aus die Möglichkeit haben wieder auf die Startseite zurückzukommen um andere Kategorien zu erforschen. Letztendlich wird bei Berührung des dafür verantwortlichen Icons, siehe Abbildung 25 (1), die URL der Startseite geladen.

[76] `https://developer.android.com/reference/android/webkit/WebView.html` zuletzt aufgerufen am 11.11.2016 um 13:33
[77] `https://developer.apple.com/reference/webkit/wkwebview` zuletzt aufgerufen am 11.11.2016 um 13:34

4 Evaluierung der Prototypen am laufendem Betrieb

Als Basis für die folgende Evaluierung dienen einerseits die Daten von Apple und Google, welche in den jeweiligen Developer-Portalen angeboten werden und andererseits die Aufzeichnungen und Statistiken von Fabric[78], welche im Zeitraum von Anfang Oktober 2016 bis Ende Februar 2017 gesammelt wurden.
Fabric ist eine externe Bibliothek, die in die Applikation integriert werden kann. Wenn Fabric eingebunden wird, fungiert sie unterstützend um bessere und vor allem stabilere Applikationen zu entwickeln. Des Weiteren kann mit Hilfe von Fabric das Verhalten von aktiven Benutzerinnen und Benutzern besser kennengelernt werden. Umgesetzt wird dies dadurch, dass Fabric, wenn es in das Projekt eingebunden ist, detaillierte Berichte über Abstürze der Applikation liefert. Diese können mit individuellen Daten erweitert werden und geben Auskunft an welcher Stelle im Quellcode schlussendlich ein Fehler aufgetreten ist. Mit Hilfe dieser Informationen kann die Fehlerquelle schnell lokalisiert und eine Lösung implementiert werden[79].
Mit dem „Answer-Kit“ bietet Fabric die Möglichkeit Aktionen von Benutzerinnen und Benutzern zu tracken. Werden diese Erkenntnisse ausgewertet, kann festgestellt werden, welche Funktionen am beliebtesten sind. Dementsprechend könnten diese ausgebaut oder andere Funktionalitäten überarbeitet und somit verbessert werden[80].

Die Prototypen wurden einerseits mit der Möglichkeit zur Protokollierung der Programmabstürze und andererseits mit einem rudimentären Tracking der einzelnen Aktionen ausgestattet. Getrackt werden dabei lediglich die Aktionen, die von den Benutzerinnen und Benutzern ausgeführt werden. Aktionen als solche sind bei der iOS-Version zum Beispiel das Abrufen eines zufälligen Artikels, die Anzeige von in der Nähe befindlichen Artikeln oder das Starten der Applikation anhand einer Benachrichtigung.
Herausgestellt hat sich hierbei, dass Benutzerinnen und Benutzer, welche ein iOS Gerät besitzen, in erster Linie bestrebt sind neue und unbekannte Informationen zu

[78] `https://get.fabric.io/` zuletzt aufgerufen am 22.02.2016 um 17:34
[79] `https://docs.fabric.io/android/crashlytics/overview.html` zuletzt aufgerufen am 22.02.2016 um 17:49
[80] `https://docs.fabric.io/android/answers/overview.html` zuletzt aufgerufen am 22.02.2016 um 17:50

erhalten. Denn das Abrufen eines zufälligen Artikels wird mit Abstand am häufigsten durchgeführt. Genauer gesagt wurden im Evaluierungszeitraum 262 Aufrufe dieser Funktion aufgezeichnet. Neben dem zufälligen Artikel ist auch der Artikel des Monats 137 mal abgerufen worden. Insgesamt wurde auch 176 mal nach Artikeln in der Nähe gesucht und 127 mal die textbasierte Suche aufgerufen. Neben dem gezielten Abrufen von Artikeln wurde auch die Navigation zur Startseite des Austria-Forums 178 mal aufgezeichnet. Werden allerdings jene Aktionen betrachtet, welche nicht für das Laden eines Artikels verantwortlich sind, ist zu erkennen, dass Benutzerinnen und Benutzer, wie oben bereits erwähnt, vor allem daran interssiert sind Informationen zu erhalten. Denn Funktionen wie Lesezeichen aufrufen oder hinzufügen, einen Artikel teilen oder weitere Lizenzinformationen über diesen abzurufen wurden zusammengezählt lediglich 81 mal im gesamten Evaluierungszeitraum durchgeführt.
Die iOS-Version des Prototypen wurde 26 mal aufgrund einer Benachrichtigung gestartet. An dieser Stelle muss für die Zukunft weiter analysiert und geklärt werden, ob die Ursache für diese geringe Anzahl an der eigentlichen Funktion oder daran liegt, dass die passive Suche nach Artikeln in der Nähe nicht aktiviert wurde, denn diese ist grundsätzlich nach Erstinstallation deaktiviert. Ein weiterer Auslöser für diese niedrige Anzahl könnte auch die Tatsache sein, dass, wie in Kapitel 2.1 bereits erwähnt, ein Großteil der Benutzerinnen und Benutzer nicht auf Benachrichtigungen reagiert.

Für Android wurden die Aktionen der Benutzerinnen und Benutzer in drei Kategorien aufgeteilt: „inApp Actions“, „inApp Navigation“ und „Start From Push “. Erstere beschreiben dabei Ereignisse bei denen Benutzerinnen und Benutzer auf dem gleichen Bildschirm verweilen. Darunter fällt zum Beispiel das Anfordern eines zufälligen Artikels, das Laden des Artikels des Monats, die Navigation zum Startbildschirm oder das Hinzufügen eines Lesezeichens. „inApp Navigation“ beschreibt jene Aktionen, bei denen Benutzerinnen und Benutzer eine neue Activity und somit ein neues Programmfenster starten. Dazu zählen zum Beispiel das Suchen nach Artikeln, die Anzeige von Artikeln in der Nähe oder das Öffnen von gesetzten Lesezeichen. „Start From Push“ wird dann protokolliert wenn Benutzerinnen und Benutzer die Applikation aufgrund einer Benachrichtigung starten.
Hier geht der Trend klar zu der Kategorie „inApp Navigation“. Mit 710 aufgezeichneten Ereignissen liegt diese klar vor der „inApp Actions“ Kategorie, welche

357 Aufrufe aufweist. Grundsätzlich kann nicht eindeutig gesagt werden weswegen Benutzerinnen und Benutzer deutlich mehr Aktionen auf demselben Bildschirm bevorzugen. Jedoch untersteicht dies die Aussagen aus Kaptiel 3.3.1 in welchem über die „Lazy User Theory“ und darüber geschrieben worden ist, dass Benutzerinnen und Benutzer den Weg des geringsten Widerstandes wählen. Denn Aktionen aus der „inApp Navigation“ Kategorie sind alle vom Startbildschirm aus durchführbar. Die „Start From Push“Kategorie ist mit drei aufgezeichneten Ereignissen sehr schwach vertreten. Dies besagt, dass auch die Android-Version des Prototypen sehr selten aufgrund einer Benachrichtigung gestartet wurde. Zwar steht diese niedrige Anzahl wieder für die in Kapitel 3.3.1 erwähnten Punkte, trotzdem muss dies für die Zukunft ebenfalls geprüft werden, ob eine Verbesserung dahingehend möglich ist.
Zusammenfassend kann auf Basis der Daten von Fabric und dessen monatlichen Berichten gesagt werden, dass die aktuelle Anzahl an aktiven Benutzerinnen und Benutzern zwischen 30 und 40 pro Monat liegt.

Bei der Analyse der Daten der jeweiligen App-Stores sind vor allem jene von Apple interessant. Hier können die sogenannten Impressionen betrachtet werden, von denen insgesamt 15375 verzeichnet wurden. Diese geben an, wie oft die Applikation an irgendeiner Stelle im App-Store gesehen wurde.[81] Davon waren 61 ausschlaggebend genug, sodass sich Benutzerinnen und Benutzer dazu entschlossen haben den Prototypen herunterzuladen. Dies führt zur Annahme, dass die Beschreibung sowie die gewählten Screenshots bzw. die Reihenfolge derer eventuell überdacht werden muss. Auf Android gibt es mit aktuellem Stand 71 Installationen. 20 von diesen wurden allerdings bereits wieder entfernt.

[81] `https://developer.apple.com/app-store/app-analytics/` zuletzt aufgerufen am 22.02.2016 um 19:45

5 Diskussion

Der resultierende Prototyp bietet viele Wege der Informationsgewinnung und setzt dabei auf die Möglichkeiten und speziellen Funktionen von mobilen Geräten. Es darf aber nicht vergessen werden, dass gewisse Features auch mit kleinen Nachteilen behaftet sind. Diese werden nun diskutiert und erläutert. Im Rahmen dieser Masterarbeit wird die Behauptung aufgestellt, dass ein zufälliger Artikel ein guter Weg ist, um Informationen, welche für Benutzerinnen und Benutzer gänzlich unbekannt sind, bereitzustellen. Ebenfalls können auf diesem Wege Informationen näher gebracht werden, welche unter Umständen niemals bestimmte Benutzerinnen und Benutzer erreicht hätten. Ein Gegenargument könnte sein, dass es sich hierbei lediglich um eine zufällige Information handelt und nicht im Interesse des/ der Lesenden ist und ohnehin gleich wieder vergessen wird. Genau aus diesem Grund gibt es im resultierenden Prototyp die Möglichkeit, dass Benutzerinnen und Benutzer eine Kategorie wählen können, aus der der zufällige Artikel stammt. Auf diese Art und Weise wird der absolute Zufallsfaktor etwas reduziert und Benutzerinnen und Benutzer erhalten dadurch Informationen, welche aus deren Interessensgebieten sind.
Ein weiterer, zu diskutierender Punkt betrifft das Feature der passiven Informationsgewinnung. Trotz der praktischen und nützlichen Anwendungsszenarien, darf nicht vergessen werden, dass dieses Feature, wenn in Verwendung, die aktuelle Position der Benutzerinnen und Benutzer durchgehend benötigt und diese regelmäßig an den Server sendet. Bei genauerer Überlegung könnte im gleichen Zuge die Frage nach der Privatsphäre gestellt werden, da Benutzerinnen und Benutzer stetig ihren Aufenthaltsort preisgeben. Genau aus diesem Grund ist es wichtig, dass diese sensiblen Daten mit äußerster Vorsicht behandelt werden. Diese sollen und werden lediglich für die Berechnung der Distanz zwischen dem Ort des/ der Benutzers/ Benutzerin und den vorhandenen Artikeln verwendet. Diesen Kompromiss müssen Benutzerinnen und Benutzer eingehen, wenn sie das Feature in Anpruch nehmen wollen. Applikationen können zwar versprechen, dass die Daten nicht für andere Zwecke missbraucht werden - eine Kontrolle, dass diese nicht missbraucht werden, gibt es leider keine für Benutzerinnen und Benutzer. Deswegen ist es ebenfalls wichtig und betrifft das Vertrauen zwischen Anbieter/ Anbieterin und Anwender/ Anwenderin, dass die Daten nur so verwendet werden, wie es in der Beschreibung angeführt ist. Stimmen Benutzerinnen und Benutzer der Nutzung ihrer Position zu, haben sie die

Möglichkeit Informationen und Artikel zu erhalten, welche direkt mit ihrer unmittelbaren Umgebung zu tun haben. Unter anderem könnten sie so auf interessante Orte, Gebäude oder Ereignisse in ihrer Nähe aufmerksam gemacht werden, über die sie sonst nie Bescheid gewusst hätten. Diese Erkenntnis könnte Benutzerinnen und Benutzer sogar überraschen. Die Anzeige von Artikeln in der Nähe könnte aber auch bestimmte Benutzerinnen und Benutzer bis zu einem gewissen Grad täuschen. Grund für diese Behauptung ist die Tatsache, dass das Ergebnis dieser Suche limitiert ist. Wenn Artikel oder Informationen nicht mit den passenden Metadaten versehen sind, werden diese nicht beachtet und auch nicht angezeigt und limitieren dadurch das Ergebnis der Suche. Für die Berechnung der Distanz werden Werte für Längengrad und Breitengrad benötigt. Artikel, die nicht damit ausgestattet wurden, werden bei der Suche nicht berücksichtigt. Dies führt dazu, dass Benutzerinnen und Benutzer eventuell glauben, dass sich gerade nicht viel Interessantes in ihrer Nähe befindet, obwohl es sehr wohl Artikel gäbe, die für den aktuellen Ort passen würden, aber nicht mit den dafür notwendigen Metadaten ausgestattet sind.
Ein weiteres Faktum ist, dass bei der passiven Suche der Lokalisierungsdienst des jeweiligen Betriebssystems im Hintergrund arbeitet, auch wenn das Smartphone gerade nicht in Verwendung ist. Dies führt natürlich unweigerlich dazu, dass in diesem Modus etwas mehr Energie verbraucht wird. Diesen Kompromiss müssen Benutzerinnen und Benutzer aber eingehen um die Möglichkeiten des Prototypen voll auszuschöpfen.
Es ist ebenfalls bekannt, dass gewisse Entscheidungen beim Erstellen des User-Interface nicht 100-prozentig mit den vorgeschlagenen Richtlinien der jeweiligen Plattformen, Android und iOS, konform gehen. Jedoch war es das Ziel des Prototypen möglichst schnell und unkompliziert Informationen zur Verfügung zu stellen und die angebotenen Features von überall und unkompliziert abrufbar zu machen. Das ist auch ein Grund warum vorgeschlagen wird, eine native mobile Applikation mit den vorgestellten Features zu implementieren, damit Benutzerinnen und Benutzer sich die Zeit sparen, einen Browser zu öffnen und mit diesem auf manuellem Wege zur Webseite zu navigieren.
Zusammengefasst gibt es ein paar kleinere Nachteile. Aber um den Prototypen möglichst nützlich und effizient zu gestallten und eine neue Art der Informationsgewinnung zu bieten, müssen Benutzerinnen und Benutzer diese Kompromisse eingehen.

6 Zusammenfassung

Als Resümee dieser Masterarbeit kann gesagt werden, dass, Benutzerinnen und Benutzer heutzutage sehr viel Zeit mit ihrem Smartphone verbringen. Deshalb ist es wichtig, dass wenn Benutzerinnen und Benutzer nach qualitativer Information streben, sie diese auch einfach und unkompliziert erhalten können. Des Weiteren müssen Lösungen angeboten werden, die diesem Bedürfnis nachkommen. Ebenso ist es wichtig, dass die angebotenen Applikationen der heutigen Zeit und dem Verhalten der gegenwärtigen Gesellschaft gerecht werden. Wir leben in einem Informationszeitalter und in solch einer digitalen Welt muss alles sofort und ohne Umwege verfügbar sein. Dementsprechend muss auch das angebotene Produkt diese Schnelligkeit anbieten. Genau deswegen muss der Weg zur Information für Benutzerinnen und Benutzer möglichst rasch vonstatten gehen und mit geringstem Aufwand verbunden sein. Basierend auf den Ergebnissen der Evaluierung von mobilen Applikationen für Nachschlagewerke hat sich herausgestellt, dass Information auch mittels Benachrichtigungen bereitgestellt werden soll. Da dieses Verhalten aber in keinem voll ausreizenden Wege beobachtet worden ist, ist festgestellt worden, dass Bedarf nach solch einer Möglichkeit existiert. Deshalb muss solch ein Feature umgesetzt werden. Damit Benutzerinnen und Benutzern Information nicht aufgezwungen wird, wurde ebenfalls entschieden, dass Möglichkeiten und Wege bereitgestellt werden müssen, mit denen Benutzerinnen und Benutzer das Verhalten der Applikation beeinflussen können. Durch diese Einstellungen haben Benutzerinnen und Benutzer Kontrolle über die Vorgehensweise dieser Funktionalität und können selbst entscheiden, wann und wie oft eine Interaktion mit der Applikation auftritt.

Diese Masterarbeit hat nach intensiver Literaturstudie und heuristischer Evaluation durch Experten/ Expertinnen eine Anzahl an Features für mobile Applikationen für Enzyklopädien identifiziert. Diese wurden implementiert und getestet. Es wird vorgeschlagen diese in Zukunft bei Applikationen dieser Art zu verwenden. Für die Zukunft können ebenfalls noch weitere Gedanken und Überlegungen durchgeführt werden, um die User-Experience für Benutzerinnen und Benutzer noch weiter zu optimieren. Obwohl der Prototyp das Prinzip des geringsten Widerstandes befolgt, können zusätzlich noch weitere Usability-tests durchgeführt werden. Mit diesen können Vor- und Nachteile gegenüber des im Rahmen dieser Arbeit umgesetzten Ansatzes im Vergleich zu der Verwendung von typischen Android und iOS spezifischen

User-Interface Elementen herausgefunden werden. Die Fragestellung dahingehend könnte lauten: Ist es besser den Aufwand zu minimieren oder den Benutzerinnen und Benutzer ein gewisses Vertrautheitsgefühl zu vermitteln?

7 Literaturverzeichnis

Literatur

Rachel Appel. Modern apps : Mobile web sites vs. native apps vs. hybrid apps, 2016. Retrieved on the 18.08.2016 um 20:22 from https://msdn.microsoft.com/en-us/magazine/dn818502.aspx?f=255&MSPPError=-2147217396.

Zach Atkin. Mobile-optimized vs responsive websites, 2015. Retrieved on the 19.08.2016 um 09:53 from https://www.atilus.com/mobile-optimized-vs-responsive-websites/.

Nick Babich. Golden rules of user interface design 10 usability heuristics for user interface design, 2016. Retrieved on the 16.08.2016 um 20:11 from https://uxplanet.org/golden-rules-of-user-interface-design-19282aeb06b#.e2krzq5iz.

George Deglin. When will web push be supported in ios?, 2015. Retrieved on the 26.08.2016 um 11:27 from https://onesignal.com/blog/when-will-web-push-be-supported-in-ios/.

Lisa Eadicicco. Americans check their phones 8 billion times a day, 2015. Retrieved on the 26.08.2016 um 10:39 from http://time.com/4147614/smartphone-usage-us-2015/.

eMarketer. Growth of time spent on mobile devices slows, 2015. Retrieved on the 26.08.2016 um 11:45 from http://www.emarketer.com/Article/Growth-of-Time-Spent-on-Mobile-Devices-Slows/1013072.

eMarketer; AP. Number of smartphone users worldwide from 2014 to 2019 (in millions), August 2015. Retrieved on the 14.10.2016 um 01:22 from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

Sabine Feierabend, Theresa Plankenhorn, and Thomas Rathgeb. Jim 2015 jugend, information, (multi-) media basisstudie zum medienumgang 12- bis 19-jähriger in deutschland. *Medienpädagogischer Forschungsverbund Südwest*, November 2015. http://www.mpfs.de/fileadmin/files/Studien/JIM/2015/JIM_Studie_2015.pdf.

Carolyn Gregoire. You probably use your smartphone way more than you think. *The Huffington Post*, November 2015. Retrieved from http://www.huffingtonpost.com/.

Sebastian Hammelehle. Beschleunigung das alles berherrschende monster. *Der Spiegel Online*, July 2013. Retrieved from http://www.spiegel.de.

Paul Hannan and Declan Sciolla-Lynch and Jeremy Hodge and Paul Withers and Tim Tripcony. *XPages Extension Library: A Step-by-Step Guide to the Next Generation of XPages Components*. IBM Press, Mai 2012.

Marijan Keleva. Windows phone 5% marktanteil in österreich. http://marijanbloggt.at/2015/08/windows-phone-5-marktanteil-in-oesterreich/121661, 2016. Retrieved on the 30.07.2016 um 17:06 from http://marijanbloggt.at/2015/08/windows-phone-5-marktanteil-in-oesterreich/121661.

André Kramer. Endgültiges aus für gedruckten brockhaus. 2014. Retrieved on the 18.08.2016 um 10:36 from http://www.heise.de/newsticker/meldung/Endgueltiges-Aus-fuer-gedruckten-Brockhaus-2293661.html.

Marco Kuhrmann. Prototyping, 2012. Retrieved on the 07.10.2016 um 08:50 from http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Vorgehensmodell/Prototyping/index.html.

Manuela Lenzen. Schluss mit dem kantinengang! *Frankfurter Allgemeine Zeitung*, September 2012. Retrieved from http://www.faz.net.

Christoph Neidhart. Digitalisierung in japan sklaven ihrer smartphones. *Sueddeutsche Zeitung*, Februar 2015. Retrieved from http://www.sueddeutsche.de.

Nils Neubauer. Mit user-onboarding-design überzeugen: Neue nutzer sollen es einfach haben mit user-onboarding-design überzeugen: Neue nutzer sollen es einfach haben, 2016. Retrieved on the 15.08.2016 um 17:29 from http://t3n.de/magazin/user-onboarding-design-neue-nutzer-239013/.

Paul Neuhold, Martin Ebner, and Hermann Maurer. Mobile applications for encyclopedias. 2017. Eingereicht zur Veröffentlichung.

Caitlin O'Connell. The inside view: How consumers really feel about push notifications, 2016. Retrieved on the 26.08.2016 um 13:35 from http://info.localytics.com/blog/the-inside-view-how-consumers-really-feel-about-push-notifications.

G. Pomberger, W. Pree, and A. Stritzinger. Methoden und werkzeuge für das prototyping und ihre integration. *Institut für Wirtschaftsinformatik, J.K.-Universität Linz*. Gelesen am 07.10.2016 um 09:09 auf http://cms.uni-salzburg.at/fileadmin/multimedia/SRC/docs/publications/J002.pdf.

Witold Pryjda. Os-anteile: Android dominiert, ios holt auf, windows schwächelt, 2016. Retrieved on the 30.07.2016 um 17:15 from http://winfuture.de/news,85309.html.

Brandy Shaul. ios push notifications have a 41 Retrieved on the 26.08.2016 um 12:24 from http://www.adweek.com/socialtimes/ios-push-notifications-have-a-41-opt-in-rate-infographic/641517.

Christian Steinbauer and Aichholzer Thomas. Prototyping. *Alpen-Adria Universität Klagenfurt*, Mai 2003. Retrieved on the 07.10.2016 um 09:09 from wwwu.edu.uni-klu.ac.at/taichhol/_neu_/?show=load&id=56.

Jason Summerfield. Mobile website vs. mobile app: Which is best for your organization?, 2016. Retrieved on the 18.08.2016 um 10:36 from https://www.atilus.com/mobile-optimized-vs-responsive-websites/.

Marianne Sweeny. How many google searchers go to page two of their search results?, 2011. Retrieved on the 18.08.2016 um 22:20 from https://www.quora.com/How-many-Google-searchers-go-to-page-two-of-their-search-results.

Franck Tétard and Mikael Collan. Lazy user theory: A dynamic model to understand user selection of products and services. *42nd Hawaii International Conference on System Sciences, 978-0-7695-3450-3/09*, 2009.

Ueli Wahli, Miguel Gomes, Brian Hainey, Ahmed Moharram, Juan Pablo Napoli, Marco Rohr, Henry Cui, Patrick Gan, Celso Gonzalez, Pinar Ugurlu, and Lara Ziosi. *Rational Application Developer V7.5 Programming Guide*. IBM Redbooks, Juni 2009. http://www.redbooks.ibm.com/redbooks/pdfs/sg247672.pdf.

Rob Weatherhead. Say it quick, say it well – the attention span of a modern internet consumer. *The Guardian*, February 2014. Retrieved from https://www.theguardian.com/.

Jan Weddehage. Mobile user experience: Der erste eindruck zählt, 2016. URL https://entwickler.de/online/ux/mobile-user-experience-erste-eindruck-zaehlt-196762.html. Retrieved on the 15.08.2016 um 17:38 from https://entwickler.de/online/ux/mobile-user-experience-erste-eindruck-zaehlt-196762.html.

A Austria Forum API Dokumentation

`getRandomPage(String[] categories)`

Debug-Url:
`http://localhost:8080/JSON-RPC`

Live-Url:
`http://austria-forum.org/JSON-RPC`

Description:
Returns information of a single WikiPage which is selected by random. There is a whitelist on the server with the name "Kategorien.txt" with all categories which shall be taken into consideration. If desired, categories can be passed alongside the request. Then only the given categories will be taken into account while selecting an article at random and creating a response with it.

Example Request:

id	To identify the calling request
method	The name of the called webservice on the server
params	The arguments for the called webservice

Code - select an Article with specified categories:

```
{
    "id":10004,
     "method":"search.getRandomPage",
     "params":[["AEIOU", "Wissenssammlungen"]]
}
```

Code - select an Article from all categories:

```
{
    "id":10004,
     "method":"search.getRandomPage",
     "params":[null]
}
```

Example Response Success:

id	To identify the calling request
result	Contains the requested information

Result Node:

javaClass (jc)	Specifies which class was serialized into a JsonString
map	A hashmap with all information about the given page
Content of the given map Element: (If the server doesn't find a page only the ResultCode and ResultDescription are given in the response to provide an meaningful error)	
ResultCode	Resultcode of the Request

ResultDescription	Result description of the request
url	The url which points to the found article (optional)
name	The name of the found article.(optional)
title	The title of the found article, use this to present it to the user.(optional)
license	A map containing further information about the license of the article. (optional)

License:

javaClass	Specifies which class was serialized into a JsonString
css	The value which will be used in order to parse the correct license logo.
id	The id of the given license
title	A human readable title of the license
url	An url where you find more information about the given license.

Code - Response Success:


```
{
  "result": {
    "javaClass": "java.util.HashMap",
    "map": {
      "license": {
        "javaClass": "java.util.HashMap",
        "map": {
          "css": "ccbysa",
          "id": "CCBYSA30",
          "title": "CC BY-SA 3.0",
          "url": "https://creativecommons.org/licenses/by-sa/3.0/"
        }
      },
      "ResultDescription": "Success",
      "name": "AustriaWiki/Gerhard_Schwarz_(Journalist)",
      "title": "Gerhard Schwarz (Journalist)",
      "ResultCode": "0",
      "url": "http://austria-forum.org/af/AustriaWiki/Gerhard_Schwarz_
%28Journalist%29"
    }
  },
  "id": 10004
}
```


Code - Response Failure:


```
{
  "result": {
    "javaClass": "java.util.HashMap",
    "map": {
      "ResultDescription": "Leider ist ein Fehler aufgetreten",
      "ResultCode": "-1"
    }
  },
  "id": 10004
}
```

`getArticleFromMonthlyPool(int notUsed, String desiredMonth, String desiredYear)`

Debug-Url:
`http://localhost:8080/JSON-RPC`

Live-Url:
`http://austria-forum.org/JSON-RPC`

Description:
Returns a map with information about an article chosen from the WikiPage "Neues und Mehr". If **@param**{desiredMonth} and **@param**{desiredYear} are set with a meaningful value for year and month, the webservice will than pick a page from that given time. If those parameters are "notset" then a page from the current year and month will be returned.
Articles from the Category "web-books/" will be excluded for now. But for the future it is planed to activate them.
The used Url Pattern for finding those Articles is:
"Neuigkeiten/{currentYear}/{currentMonthAsString}_{currentYear}"

Example Request:

id	To identify the calling request
method	The name of the called webservice on the Server
params	The arguments for the called sebservice

Code - select an article from the given month and year:

```
{
   "id":10004,
    "method":"search.getArticleFromMonthlyPool",
    "params":[1337,"desiredMonth","desiredYear",lookBack]
}
```

Code - select an article from the current month and year:

```
{
   "id":10004,
    "method":"search.getArticleFromMonthlyPool",
    "params":[1337,"notset","notset",1]
}
```

Example Response Success:

id	To identify the calling request
result	Contains the requested information

Result Node:

javaClass	Specifies which class was serialized into a JsonString
map	A hashmap with all information about the given page, serialized into json
Content of the given map Element: (If the server doesn't find a page only the ResultCode and ResultDescription are given in the response to provide an meaningful error)	
ResultCode	Resultcode of the Request

ResultDescription	Result description of the Request
url	The url which points to the found article (optional)
name	The name of the found article.(optional)
title	The title of the found article, use this to present it to the user.(optional)
license	The license for the corresponding article.(optional)

License:

javaClass	Specifies which class was serialized into a JsonString
css	The value which will be used in order to parse the correct license logo.
id	The id of the given license
title	Human readable title of the license
url	An url where you find more information about the given license.

Example Response Success:


```
{
  "result": {
    "javaClass": "java.util.HashMap",
    "map": {
      "ResultDescription": "Success",
      "name": "Geography/Asia/Pakistan/Special_Information/Khewra_Salt_Mines",
      "title": "Khewra Salt Mines",
      "ResultCode": "0",
      "url": "http://austria-
forum.org/af/Geography/Asia/Pakistan/Special_Information/Khewra_Salt_Mines"
    }
  },
  "id": 10004
}
```


Example Response Failed:


```
{
   "result":{
      "javaClass":"java.util.HashMap",
      "map":{
         "ResultDescription":"In diesem Monat gibt es noch keine neuen Artikel",
         "ResultCode":"-1"
        "site": "Tried to access Neuigkeiten/2016/Mai_2016",
      }
   },
   "id":10004
}
```

`getAllPagesInRange(double latitude, double longitude, double range, int maxLength)`

Debug-Url:
`http://localhost:8080/JSON-RPC`

Live-Url:
`http://austria-forum.org/JSON-RPC`

Description:
Returns all articles which are found in a specific radius around the current position of the user. The current position of user is based on the given latitude and longitude. On success a list of **@param**{maxLength} is returned. The ascending order of the list is based on the distance between the users current position and the location of the relevant article. Each element of the list contains information about the corresponding article.

Example Request:

id	To identify the calling request
method	The name of the called webservice on the Server
params	The arguments for the called webservice

Code:
```
{
   "id":10004,
    "method":"search.getAllPagesInRange",
    "params":[latitude, longitude, range, maxLength]
}
```

Example Response Success:

id	To identify the calling request
result	Contains the requested information

Result Node:

javaClass	Specifies which class was serialized into a JsonString
list	An ArrayList serialized into a JsonArray. ***Is empty when no articles are found.***
map (element of list-JsonArray)	A aashmap with all information about the given page, serialized into json
Content of the given map element:	
url	The url which points to the found article
page	The name of the found article.
title	The title of the found article, use this to present it to the User.
license	The license for the corresponding article. (optional)

distance	The distance to the users position given in meters.

License:

javaClass	Specifies which class was serialized into a JsonString
css	The value which will be used in order to parse the correct license logo.
id	The id of the given license
title	A human readable title of the license
url	An url where you find more information about the given license.

Example Response Failed:

```
{
   "result": {
    "javaClass": "java.util.ArrayList",
    "list": []
  },
  "id": 10004
}
```

Example Response Success:

```
{
  "result": {
    "javaClass": "java.util.ArrayList",
    "list": [
      {
        "javaClass": "java.util.HashMap",
        "map": {
          "distance": 537,
          "page": "AustriaWiki/Herz-Jesu-Kirche_(Graz)",
          "title": "Herz-Jesu-Kirche (Graz)",
          "url": "http://austria-forum.org/af/AustriaWiki/Herz-Jesu-Kirche_
                  %28Graz%29"
        }
      },
      {
        "javaClass": "java.util.HashMap",
        "map": {
          "distance": 539,
          "page": "AustriaWiki/Hallerschloss_(Graz)",
          "title": "Hallerschloss (Graz)",
          "url": "http://austria-forum.org/af/AustriaWiki/Hallerschloss_%28Graz
                  %29"
        }
      }
    ]
  },
  "id": 10004
}
```

getPageInfo(String[] urls)

Debug-Url:
http://localhost:8080/JSON-RPC

Live-Url:
http://austria-forum.org/JSON-RPC

Description:
This function returns information about the requested urls. If the given values in the url array are pointing to an existing WikiPage of the austria-forum, information about those pages will be returned.

Example Request:

id	To identify the calling request
method	The name of the called webservice on the server
params	The arguments for the called webservice

Code:

```
{
   "id":10004,
    "method":"search.getPageInfo",
    "params":[["http://austria-
forum.org/af/Wissenssammlungen/Essays/Vermischtes/Das_Gl%C3%BCcksschwein"]]
}
```

Example Response Success:

id	To identify the calling request
result	Contains the requested information

Result Node:

javaClass	Specifies which class was serialized into a JsonString
list	An arrayList serialized into a JsonArray.
map (element of list-JsonArray)	A hashmap with all information about the given page, serialized into json
Content of the given map Element:	
ResultCode	The result code for the request
query	The query which was used to look for information (url)
url	The url which points to the found article (optional)
name	The name of the found article.(optional)
title	The title of the found article, use this to present it to the user.(optional)
license	The license for the corresponding article. (optional)

License:

javaClass	Specifies which class was serialized into a JsonString
css	The value which will be used in order to parse the correct license logo.
id	The id of the given license
title	A human readable title of the license
url	An url where you find more information about the given license.

Example Response Success:

```
{
  "result": {
    "javaClass": "java.util.ArrayList",
    "list": [
      {
        "javaClass": "java.util.HashMap",
        "map": {
          "query": "url_of_wiki_page",
          "name": "Wissenssammlungen/Essays/Ökologie/Waldökologie_Dürrenstein",
          "title": "Waldökologie Dürrenstein",
          "ResultCode": "0",
          "url": "http://austria-forum.org/af/Wissenssammlungen/Essays/
                  %C3%96kologie/Wald%C3%B6kologie_D%C3%BCrrenstein"
        }
      }
    ]
  },
  "id": 10004
}
```

Example Response Failed:

```
{
  "result": {
    "javaClass": "java.util.ArrayList",
    "list": [
      {
        "javaClass": "java.util.HashMap",
        "map": {
          "query": "http://austria-forum.org/af/Wissenssammlungen/Essays/
                  %C3%96kologie/Wald%C3ologie_D%C3%BCrrenstein",
          "ResultCode": "-1"
        }
      }
    ]
  },
  "id": 10004
}
```

`findPagesMobile(String query, int maxLength)`

Debug-Url:
`http://localhost:8080/JSON-RPC`

Live-Url:
`http://austria-forum.org/JSON-RPC`

Description:
The function scans austria-forums archive for the given query and returns pages which fits the searched string. This webservice is a slightly modified version of the already existing findPages. It was extended in order to provide additional information about the found pages to fit the needs of the mobile application.

Example Request:

id	To identify the calling request
method	The name of the called webservice on the server
params	The arguments for the called webservice

Code:

```
{
   "id":10004,
    "method":"search.findPagesMobile",
    "params":["Tesla","2"]
}
```

Example Response Success:

id	To identify the calling request
result	Contains the requested information

Result Node:

javaClass	Specifies which class was serialized into a JsonString
list	An ArrayList serialized into a JsonArray. **Is empty when no articles are found.**
map (element of list-JsonArray)	A hashmap with all information about the given page, serialized into json
Content of the given map element:	
url	The url which points to the found article
page	The name of the found article.
title	The title of the found article, use this to present it to the user.
score	Between 0 and 100. Represents the relevance of an article. The higher the score the more the article fits the query.
license	The license for the corresponding article. (Optional)

License:

javaClass	Specifies which class was serialized into a JsonString
css	The value which will be used in order to parse the correct license logo.
id	The id of the given license
title	A human readable title of the license
url	An url where you find more information about the given license.

Example Response Success:


```
{
   "result": {
    "javaClass": "java.util.ArrayList",
    "list": [
      {
        "javaClass": "java.util.HashMap",
        "map": {
          "score": 100,
          "page": "Wissenssammlungen/Biographien/Tesla,_Nikola",
          "title": "Tesla, Nikola",
          "url": "http://austria-
      forum.org/af/Wissenssammlungen/Biographien/Tesla%2C_Nikola"
        }
      },
      {
        "javaClass": "java.util.HashMap",
        "map": {
          "score": 100,
          "page": "AEIOU/Tesla,_Nikola",
          "title": "Tesla, Nikola",
          "url": "http://austria-forum.org/af/AEIOU/Tesla%2C_Nikola"
        }
      }
    ]
  },
  "id": 10004
}
```


Example Response Failed:


```
{
 "result": {
    "javaClass": "java.util.ArrayList",
    "list": []
  },
  "id": 10004
}
```

Possible License Values (07. February 2017):

ResponseCode id	Title	Description Url:
AF	Austria-Forum - Alle Rechte vorbehalten	http://austria-forum.org/af/Lizenzen/Austria-Forum
PD	Gemeinfrei - kein bekanntes Urheberrecht	https://creativecommons.org/about/pdm/
CC0	Creative Commons Kein Urheberrechtsschutz	https://creativecommons.org/publicdomain/zero/1.0/
CCBY40	Creative Commons Namensnennung 4.0 International	"https://creativecommons.org/licenses/by/4.0/
CCBY30	Creative Commons Namensnennung 3.0 Unported	https://creativecommons.org/licenses/by/3.0/
CCBY20	Creative Commons Namensnennung 2.0 Generic	https://creativecommons.org/licenses/by/2.0/
CCBYSA40	Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International	https://creativecommons.org/licenses/by-sa/4.0/
CCBYSA30	Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 3.0 Unported	https://creativecommons.org/licenses/by-sa/3.0/
CCBYSA25	Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 2.5 Generic	https://creativecommons.org/licenses/by-sa/2.5/
CCBYSA20	Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 2.0 Generic	https://creativecommons.org/licenses/by-sa/2.0/
CCBYND40	Creative Commons Namensnennung - Keine Bearbeitungen 4.0 International	https://creativecommons.org/licenses/by-nd/4.0/
CCBYND30	Creative Commons Namensnennung - Keine Bearbeitung 3.0 Unported	https://creativecommons.org/licenses/by-nd/3.0/
CCBYND20	Creative Commons Namensnennung- Keine Bearbeitung 2.0 Generic	https://creativecommons.org/licenses/by-nd/2.0/
CCBYNC40	Creative Commons Namensnennung - Nicht kommerziell 4.0 International	https://creativecommons.org/licenses/by-nc/4.0/
CCBYNC30	Creative Commons Namensnennung - Nicht kommerziell 3.0 Unported	https://creativecommons.org/licenses/by-nc/3.0/
CCBYNC20	Creative Commons Namensnennung - Nicht kommerziell 2.0 Generic	https://creativecommons.org/licenses/by-nc/2.0/
CCBYNCSA40	"Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International	https://creativecommons.org/licenses/by-nc-sa/4.0/
CCBYNCSA30	Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 3.0 Unporte	https://creativecommons.org/licenses/by-nc-sa/3.0/
CCBYNCSA20	Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 2.0	https://creativecommons.org/licenses/by-nc-sa/2.0/

	Generic	
CCBYNCND40	Creative Commons Namensnennung - Nicht kommerziell - Keine Bearbeitungen 4.0 International	https://creativecommons.org/licenses/by-nc-nd/4.0/
CCBYNCND30	Creative Commons Namensnennung - Namensnennung - Nicht-kommerziell - Keine Bearbeitung 3.0 Unported	https://creativecommons.org/licenses/by-nc-nd/3.0/
CCBYNCND20	Creative Commons Namensnennung - Nicht-kommerziell - Keine Bearbeitung 2.0 Generic	https://creativecommons.org/licenses/by-nc-nd/2.0/