Plattformunabhängige App-Entwicklung für mobile Geräte
—
Grenzen und Möglichkeiten

BACHELORARBEIT

im Studiengang
Medieninformatik

des Fachbereichs
Informatik und Medien

der
Beuth Hochschule für Technik Berlin

Vorgelegt von
Daniel Morgenstern

im Sommersemester 2014

Betreuende Lehrkraft:
Prof. Dr. Simone Strippgen

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Ziel der Arbeit

1.2 Motivation

1.3 Aufbau der Arbeit

Kapitel 2

Theoretische Grundlagen

2.1 Apps für mobile Geräte

2.1.1 Mobile (native) Apps

Unter mobilen *Apps (Kurzform für engl. „Application“)* versteht man im Allgemeinen Anwendungssoftware für Tablet-Computer oder Smartphones. Im Laufe der letzten Jahre haben sich auf dem Markt für Mobilgeräte durch viele konkurierende Gerätehersteller eine Vielzahl von Smartphone- und Tablet-Betriebssystemen herausgebildet. Im Entwicklungsbereich wird in dem Zusammenhang auch von *Plattformen* gesprochen.

Zu den Plattformen mit dem höchsten Marktanteil zählen Googles Betriebssystem *Android*, *iOS* von Apple, *Microsoft Windows Phone* und *Blackberry OS* des gleichnamigen Smartphone-Herstellers Blackberry [6]. Die App-Entwicklung für diese mobilen Betriebssysteme erfolgt mehr oder weniger ähnlich und soll im Folgenden, um auf die beiden größten Vertreter einzugehen, anhand von Android beziehungsweise iOS näher beschrieben werden.

Grundsätzlich müssen auf der *Entwicklungsumgebung* die entsprechenden *SDKs* der Plattform, für die entwickelt wird, installiert sein. Diese enthalten Softwarekomponenten, die zur Entwicklung der App notwendig sind, beispielsweise Klassen, die es einem erlauben, auf native Funktionalitäten des Betriebssystems wie zum Beispiel das Adressbuch, den Benachrichtigungsmechanismus oder auch auf Hardwarekomponenten wie die Kamera, den Bewegungssensor oder das *GPS*-Modul zuzugreifen sowie die entsprechenden plattformspezifischen Oberflächenkomponenten des jeweiligen *GUI*-Toolkits zu nutzen.

Als Programmiersprache für die Android-App-Entwicklung wird Java verwendet. Das heißt, als Voraussetzung für die Entwicklung von Android-Apps ist lediglich eine geeignete Entwicklungsumgebung wie *Eclipse*, *Netbeans IDE* oder *IntelliJ IDEA* sowie eine Installation des Java- und des Android-SDK nötig. Seit 2013 bietet Google darüberhinaus die auf IntelliJ IDEA basierende und eigens für die Android-Entwicklung angepasste Entwicklungsumgebung *Android Studio* an [1], die bereits alle notwendigen Toolkits enthält. Nachdem der Code geschrieben ist, kann er kompiliert und zu einem lauffähigen Programm gebaut werden (siehe Abbildung 2.1). Anschließend kann die App in dem für die Zielplattform vorgesehenen Dateiformat ausgeliefert und auf dem Zielgerät installiert werden.

Auch der Software- und Computer-Hersteller Apple bietet mit *Xcode* eine firmeneigene Entwicklungsumgebung zur App-Entwicklung für sein mobiles Betriebssystem iOS an. Anders als Google geht der iPhone-Hersteller hier allerdings etwas restriktiver vor. So läuft die Entwicklungsumgebung Xcode, die man für die native iOS-Entwicklung benötigt, nur unter dem hauseigenen Betriebssystem *Mac OS X* und das wiederum nur auf den firmeneigenen Mac-Rechnern. So sichert sich Apple auch durch jeden Entwickler einen neuen Kunden. Ansonsten verläuft der Entwicklungsprozess bei der iOS-Entwicklung im Prinzip ähnlich zur Android-Entwicklung (siehe Abbildung 2.1). Als Programmiersprache wird *ObjectiveC* verwendet, einer um objektorientierte Elemente erweiterte Variante der Programmiersprache *C*.

Möchte ein Auftraggeber einer Software also statt seinen Kunden nur eine App für ein Betriebssystem anzubieten, einen größeren Nutzerkreis erschließen, muss die zu entwicklende App für jede Zielplattform neu programmiert, getestet und gebaut werden, da jede mobile Plattform ihre eigenen Toolkits, Bibliotheken und Programmiersprachen verwendet, was die native App-Entwicklung für potenzielle Auftraggeber zu einem sehr kostenaufwändigen Projekt werden lassen kann. Andererseits bietet die native App-Entwicklung vollständige Unterstützung der betriebssystemeigenen Funktionalitäten wie den Zugriff auf Kamera, Adressbuch, Bewegungssensoren etc. der jeweiligen Plattform, sodass ein Softwareprojekt mit solchen besonders hardware- oder betriebssystemnahnen Anforderungen die Entwicklung einer nativen (plattformspezifischen) App notwendig erscheinen lassen kann.[1]

[1] Mehr dazu in Unterabschnitt 2.2.3

2.1.2 Web-Anwendungen

Eine *Web-App (oder dt. „Web-Anwendung“)* ist eine Andwendungssoftware, die auf einem Web-Server läuft und auf die der Nutzer mittels eines Browsers zugreifen kann; also eine dynamische Website, wie man sie auch schon vor dem Aufkommen von Smartphones und modernen Tablets kannte.

Die Grundlage für die Enwicklung von Internetseiten bildet der langjährige Standard *HTML*, mit dem deren Aussehen, Inhalt und Struktur textuell beschrieben werden kann. In Kombination mit *CSS* für die modulare Gestaltung einer Website sowie *JavaScript*, einer Skriptsprache zur *DOM*-Manipulation, bietet die HTML-Spezifikation in ihrer neusten Version *(HTML5)* im Grunde alles, was für die Entwicklung einer modernen Benutzerschnittstelle am Computer notwendig ist. In Kombination mit CSS für die modulare Gestaltung einer Website sowie JavaScript, einer Skriptsprache zur DOM-Manipulation, bietet die HTML-Spezifikation in ihrer neusten Version *(HTML5)* im Grunde alles, was für die Entwicklung einer modernen Benutzerschnittstelle am Computer notwendig ist. Die Fachlogik liegt, neben den Oberflächen-Komponenten in Form von HTML-, CSS- und Javascript-Dokumenten, auf einem Webserver und verarbeitet und reagiert auf Anfragen des Clients[2]. Als Server-Technologie ist ein breites Spektrum an Programmiersprachen und Umgebungen einsetzbar[3].

Somit bietet die Entwicklung einer Web-App (abgesehen von einigen browser-spezifischen Eigenheiten) bereits eine gewisse Plattformunabhängigkeit, da jedes moderne (mobile) Betriebssystem über einen Webbrowser verfügt. Zwar müssen Entwickler in bestimmten Details bei der Erstellung des Codes auf die teilweise unterschiedliche Unterstützung (bspw. von HTML-Elementen) durch die verschiedenen Browser achten, aber darüberhinaus wird der Entwicklungsaufwand für eine Web-App nicht von der Anzahl der Zielplattformen bestimmt, da von Client-Seite aus verschiedene Browser durch die Verbreitung und Beachtung von Web-Standards weitgehend einheitliche HTML-Dokumente lesen und interpretieren können und das Backend nicht auf Clients mit unterschiedlichen Plattformen, sondern

[2] hier also des Browsers

[3] einige sind beispielsweise PHP, Java, ASP u. v. a. m.

auf Webservern liegt, deren Plattform bei der Entwicklung entweder schon bekannt oder nicht relevant ist.[4]

Obwohl es, durch damals eher im Business-Bereich verortete Internet-Handys und Palmtops, auch vor den heute üblichen mobilen Touch-Geräten bereits mobile Internetseiten gab, die speziell für die Darstellung auf kleinen Displays ausgerichtet waren, boten mit der massenhaften Verbreitung von mobilen, internetfähigen Geräten und deren (im Folgenden erläuterten) stark anwendungsorientierten Bedien-Konzepten viele herkömmliche Internet-Dienste nun auch zusätzlich eine native App für verschiedene mobile Plattformen an. So sind beispielsweise auch E-Mail-Dienste wie GMX, Web.de oder Gmail seit der Verbreitung von Smartphones und Tablets auch in Form einer eigenen App für Android und iOS vertreten, sodass der Nutzer, statt, wie von der Desktop-Computer-Nutzung gewohnt, einen anbieterunabhängigen Mail-Client zu konfigurieren, über den er seine E-Mails abruft, unter Umständen gleich die jeweilige App des E-Mail-Anbieters startet [5, 9, 10]. Das heißt, der Nutzer folgt einem geänderten Bedienungsmuster seines Mobilgeräts gegenüber der herkömmlichen Computer-Nutzung: um zu einem bestimmten Ergebnis zu gelangen (bspw. *Nachrichten lesen*) also die Frage zu beantworten, *wie* er dahin gelangt (Einen Browser öffnen und zur gewünschten Seite navigieren: www.tagesschau.de), ist es für Anwender heutiger Mobilsysteme naheliegend, gleich die passende App zu starten[5].

Dafür gibt es verschiedene mögliche Gründe. Zum Einen muss im Gegensatz zu einer Website bei der mobilen App nicht die komplette Oberfläche[6] übertragen werden, sondern lediglich die Nutzdaten,[7] was dem Nutzer ein höheres Maß an Performanz einbringt. Zum Anderen können trotz Vollbildmodus in bestimmten Fällen GUI-Elemente des Webbrowsers bei der Benutzung einer Web-Anwendung störend sein, so ist beispielsweise die Adresszeile am Rand nicht unbedingt erwünscht, wenn der Nutzer statt im Internet zu surfen dort eigentlich eine bestimmte Anwendung nutzen möchte. Ein anderes Beispiel für ein eventuell unerwünschtes Verhalten der Benutzerschnittstelle ist das der *Menü*-Taste bei Android-Geräten, die im Falle der Nutzung

[4] beispielsweise weil auch die Fachlogik plattformunabhängig mit PHP oder Java realisiert wurde

[5] hier bspw. die Tagesschau-App

[6] HTML-, CSS- und JavaScript-Dokumente sowie Grafiken

[7] also beispielsweise, um beim obigen Beispiel zu bleiben, die Nachrichten in Textform.

einer Web-Anwendung über den Browser nicht den Kontext der eigentlich benutzten Anwendung [8] anzeigt, sondern lediglich den des Browsers.

In bestimmten Fällen kann eine nützliche Funktion einer App die Offline-Nutzung sein, wenn beispielsweise durch die abgedeckten Anwendungsfälle keine Verbindung oder Synchronisation mit einem Server nötig ist. Beispiele hierfür könnten, um nur einige zu nennen, ein Taschenrechner, kleine Spiele, oder eine Bildverarbeitungs-App sein. Für diese Offline-Nutzung einer App zeichnet die Web-Anwendung ein geteiltes Bild: Zwar wurden in den letzten Jahren mehrere Methoden entwickelt, eine Web-Anwendung auch offline nutzen zu können, doch durch ihre Ausrichtung auf die Nutzung via Internet stellt die Implementierung dieser Funktionalität für Entwickler einen Zusatzaufwand dar.

Einige Möglichkeiten, eine Web-Anwendung ohne Internetverbindung nutzbar zu machen, sind beispielsweise die aus der HTML5-Spezifikation hervorgehenden Technologien *WebStorage*, ein Mechanismus zum lokalen Speichern von größeren Datenmengen in Form von Schlüssel-Wert-Paaren [3] sowie *WebSQL* bzw. *IndexedDB*, beides auf Web-Anwendungen otimierte Datenbanken-Spezifikationen, die vom *W3C* herausgegeben werden [2, 7]. Allerdings bestehen auch bei diesen Mechanismen teilweise Einschränkungen durch die Browservielfalt beziehungsweise deren Versionen. So wird IndexedDB beispielsweise nicht von Safari oder iOS unterstützt, Google Chrome muss für die Nutzung mindestens in Version 23 oder höher vorliegen, Mozilla Firefox in 10 oder höher. Auch bei WebSQL zeichnet sich ein ähnlich diffuses Bild ab: Während Chrome die Technologie ab der Version 4 und iOS ab Version 3.2 unterstützt, ist für Nutzer der Browser Firefox und Microsoft Internet Explorer die Technik gar nicht verfügbar. Lediglich WebStorage wird weitgehend von allen gängigen Browsern unterstützt [8]. Weiterhin wird die Offline-Funktionalität gegenüber der nativen App dadurch eingeschränkt, dass der Nutzer diese ohne weiteres Zutun des Entwicklers nur dann nutzen kann, wenn die entsprechende Internet-Seite im Offline-Zustand des Geräts bereits im Browser geöffnet ist, da diese nicht lokal auf dem Gerät, sondern auf einem Webserver gespeichert ist. Für vollständigen Offline-Zugriff müsste der Entwickler die komplette Website so paketieren,

[8] hier also der Website

dass der Nutzer sie – wie eine native App – von seinem Gerät aus starten und nutzen kann.[9]

Allgemein kann man sagen, dass der Zugriff auf native Funktionalitäten des Geräts respektive des Betriebssystems nicht oder nur gering unterstützt wird, sodass der geringere Enwicklungsaufwand einer solchen Web-App (siehe Abbildung 2.1) unter Umständen zu Lasten des Funktionsumfangs und der Usibility der Anwendung geht.

2.1.3 Hybride Apps

Die *Hybrid-App* verbindet Eigenschaften der Nutzung einer nativen App mit den Vorteilen der Web-Entwicklung mithilfe von Web-Technologien und entsprechenden Frameworks und löst damit beispielsweise das Problem der mangelnden Offline-Fähigkeit einer Web-App sowie deren geringe Unterstützung von plattformspezifischen oder hardware-nahen Funktionalitäten. Da jede moderne mobile Betriebssystem für Entwickler auch die Möglichkeit bietet, eine Web-View in die zu entwicklende App einzubinden, also eine GUI-Komponente, in die HTML-Seiten hineingeladen werden können, liegt der Ansatz für hybride Apps auf der Hand: Auf Entwicklungsebene wird die Anwendung als Web-App entwickelt, gleichzeitig mithilfe von entsprechenden APIs und Frameworks zur Anbindung an die native Ebene der Zielplattform in eine App für die jeweiligen Plattformen integriert, sodass auf Benutzerseite die Nutzung einer Web-Anwendung, die in puncto Funktionsumfang, Usability und Look-And-Feel einer nativen mobilen App sehr nahe kommt, möglich wird.

Dieses Vorgehen bietet unter anderem für Web-Entwickler den Vorteil, ihre bisherigen Programmierkenntnisse im Web-Bereich im Wesentlichen auch für die Entwicklung von hybriden Apps nutzen zu können. So wird in der Regel der grundlegende Teil des Codes für das Frontend, wie bei der Web-Entwicklung, in HTML in Kombination mit CSS und JavaScript geschrieben und getestet. Da allerdings auf dem mobilen Gerät für das Backend, also die Verarbeitungsinstanzen, nicht, wie bei einer herkömmlichen Web-Anwendung, ein Server mit einer entsprechenden Server-Technologie wie PHP oder ASP läuft, wird auch dieser Teil der App bei der hybriden Entwicklung meist mit JavaScript bewerkstelligt. Aber auch andere Skriptsprachen, die lokal auf einem Mobilgerät ausführbar sind, sind denkbar.

[9] siehe Unterabschnitt 2.1.3 und 2.2.3.

Anschließend muss die Anwendung für die verschiedenen Zielplattformen gebaut werden, um in das jeweilige Container-Format für Apps der verschiedenen Plattformen eingebunden werden zu können und den Zugriff auf die plattformspezifischen Toolkits durch die Cross-Platform-APIs zu ermöglichen (Abbildung 2.1). Hierfür kann es erforderlich sein, dass auf der Entwicklungsplattform die jeweiligen SDKs installiert sind, was gegenüber der Web-Entwicklung einen administrativen Mehraufwand darstellt. Eine andere Variante ist die Auslagerung des Bauprozesses auf einen externen Build-Server, beispielsweise mithilfe eines externen Web-Service eine Drittanbieters, was den Vorteil hat, die SDKs für die Zielplattformen nicht auf jedem Entwicklungsrechner verwalten zu müssen. Allerdings hat der Entwickler durch die Herausgabe des Codes an einen solchen Dienstleister nicht mehr die vollständige Kontrolle über den Code, sodass die Variante der Auslagerung des Build-Prozesses gerade für Closed-Source-Projekte tendentiell nicht in Frage kommt. Desweiteren kann der Betreiber des Build-Services unter Umständen Restriktionen bezüglich der Plattformunterstützung erteilen, wodurch eventuell eine geringere Anzahl von Zielplattformen unterstützt wird, als von Entwicklerseite gewünscht oder erfordert.[10]

2.2 Plattformunabhängige App-Entwicklung

2.2.1 Möglichkeiten des Erreichens von Plattformunabhängigkeit

2.2.2 Lösungen für die hybride App-Entwicklung

2.2.3 Entwicklung von hybriden Apps

Wie in 2.1.3 beschrieben, bildet die hybride App-Entwicklung die Schnittmenge aus der nativen App-Entwicklung und der Web-Entwicklung mithilfe von Web-Technologien und zusätzlichen Frameworks und Bibliotheken. Hier soll mit *PhoneGap* die konkrete Nutzung eines dieser Frameworks und weitere verwendete Technologien wie *Knockout* oder *JQueryMobile* erläutert werden.

Bereits für die Entwicklung von reinen Web-Anwendungen stellen neben den grundlegenden Web-Technologien HTML, CSS und JavaScript Erweite-

[10] Beispielsweise unterstützt *PhoneGap Build* in der neusten Version 3 nur nur noch die drei großen Mobilplattformen Android, iOS, und Windows Phone.

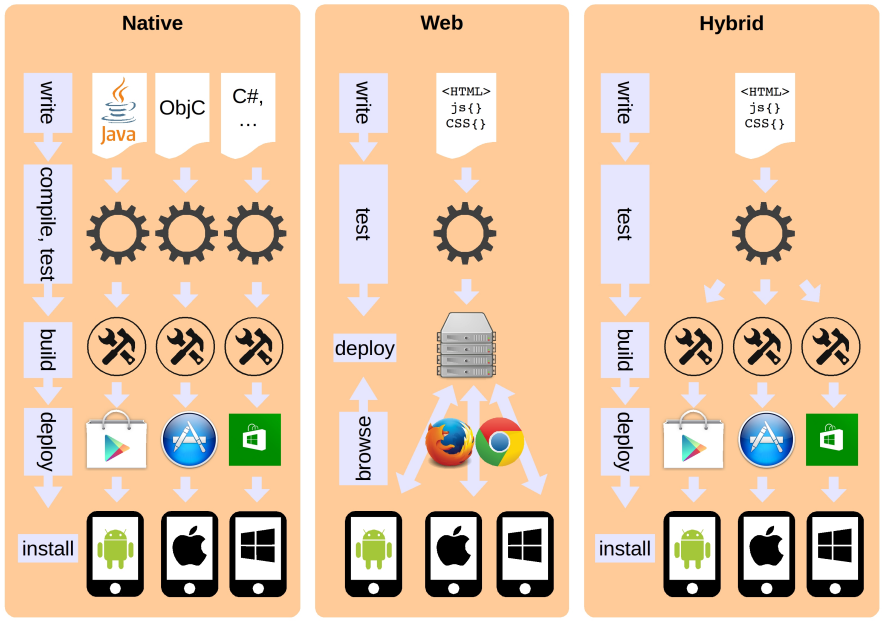

Abbildung 2.1: Entwicklungsstufen der verschiedenen Arten von Apps. Während bei der nativen App der gesamte Entwicklungszyklus einmal pro Plattform durchlaufen werden muss, verringert sich der Aufwand für die Web-App erheblich. Bei der Hybrid-App muss die Anwendung zwar einmal für jede Plattform gebaut und ausgeliefert werden um die Schnittstellen für die nativen Plattformen zu implementieren, aber der hauptsächliche Entwicklungsaufwand des Programmierens und Testens fällt aufgrund des generischen Charakters nur einmal an.

rungen wie die JavaScript-Bibliothek *JQuery* nützliche Hilfsmittel dar, die viele Funktionen gegenüber der Verwendung von „reinem“ JavaScript deutlich vereinfacht. So ist beispielsweise der Programmieraufwand für den Zugriff auf Elemente einer HTML-Seite durch JQuery wesentlich geringer als ohne die Bibliothek. Besonders deutlich wird dies an den unten aufgeführten Code-Beispielen, in denen ein Button exemplarisch die Funktionalität übernehmen soll, alle Absätze einer HTML-Seite auszublenden. Während bei herkömmlichem JavaScript für die Selektion aller Elemente die Funktion `getElementsByTagName()` aufgerufen werden muss (siehe Listing 2.2), ist bei JQuery der Zugriff auf alle Elemente eines *Tags* per Dollar(`$`)-Notation deutlich verkürzt (siehe Listing 2.1). Auch die nächste Anweisung zur Aus-

führung einer Operation für alle ausgewählten Elemente (hier: `hide()`, also *ausblenden*) fällt bei JQuery wesentlich kürzer aus, indem die Funktion noch in der selben Zeile wie der vorherigen Selektor aufgerufen werden kann (`$("p").hide();`). Bei der reinen JavaScript-Variante ist nach der Selektion aller Absätze zunächsteinmal ein Array ausgewählt, sodass, um auf den einzelnen Elementen Oparationen ausführen zu können, durch alle Elemente des Arrays iteriert und die Funktion für jedes Element aufgerufen werden muss (siehe Listing 2.2, Zeilen 6-8).

Listing 2.1: JQuery-Beispiel-Code zum Ausblenden aller Absätze [4].

```
1  <html>
2    <head>
3      <script src="libs/jquery.min.js"></script>
4      <script>
5        $(document).ready(function() {
6          $("button").click(function() {
7            $("p").hide();
8          });
9        });
10     </script>
11   </head>
12   <body>
13     <h2>This is a heading</h2>
14     <p>This is a paragraph.</p>
15     <p>This is another paragraph.</p>
16     <button>Click me</button>
17   </body>
18 </html>
```

Listing 2.2: Die selbe Funktionalität wie in Listing 2.1 mit reinem JavaScript.

```
1  <html>
2    <head>
3      <script>
4        function hideParagraphs() {
5          var paragraphs = document.getElementsByTagName("p");
6          for (i in paragraphs ) {
7            paragraphs[i].style.display = "none";
8          }
9        }
10     </script>
11   </head>
12   <body>
```

```
13      <h2>This is a heading</h2>
14      <p>This is a paragraph.</p>
15      <p>This is another paragraph.</p>
16      <button onclick="hideParagraphs()">Click me</button>
17    </body>
18  </html>
```

JQueryMobile

KnockoutJS

Phonegap

PhoneGap ist ein Framework zur hybriden App-Entwicklung von Adobe und baut auf...

PhoneGap Build

Quellenverzeichnis

Online-Quellen

[1] Xavier Ducrohet, Tor Norbye und Katherine Chou. *Android Studio: An IDE built for Android*. Mai 2013. URL: http://android-developers.blogspot.in/2013/05/android-studio-ide-built-for-android.html (siehe S. 3).

[2] Ian Hickson. *Web SQL Database*. W3C. 18. Nov. 2010. URL: http://www.w3.org/TR/2010/NOTE-webdatabase-20101118/ (siehe S. 6).

[3] Ian Hickson. *Web Storage*. W3C. 2013. URL: http://www.w3.org/TR/2013/REC-webstorage-20130730/ (siehe S. 6).

[4] *jQuery Examples*. URL: http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_hide_p (besucht am 11. 04. 2014) (siehe S. 10).

[5] *Kostenlose GMX Mail App - Ihr Postfach immer & überall dabei*. 1&1 Mail & Media GmbH. URL: http://www.gmx.net/produkte/mobile/mail-app (siehe S. 5).

[6] Ramon Llamas, Ryan Reith und Michael Shirer. *Android Pushes Past 80% Market Share While Windows Phone Shipments Leap 156.0% Year Over Year in the Third Quarter, According to IDC*. IDC Corporate USA. Nov. 2013. URL: http://www.idc.com/getdoc.jsp?containerId=prUS24442013 (siehe S. 2).

[7] Nikunj Mehta u. a. *Indexed Database API*. W3C. Juli 2013. URL: http://www.w3.org/TR/2013/CR-IndexedDB-20130704/ (siehe S. 6).

[8] *Offline - HTML5 Rocks*. Google Inc. URL: http://www.html5rocks.com/de/features/storage (siehe S. 6).

[9] Rene Ritchie. *Google releases official Gmail app for iPhone, iPad*. Nov. 2011. URL: http://www.imore.com/google-releases-official-gmail-app-iphone-ipad (siehe S. 5).

[10] *WEB.DE App - Postfach Apps für Smartphone, iPhone und iPad*. 1&1 Mail & Media GmbH. URL: https://produkte.web.de/freemail_mobile_startseite (siehe S. 5).